ANALYZING SOCIAL MEDIA CONTENTS

KANG WEI

(M.Sc., Northwestern Polytechnical University, 2009)
(B.Sc., Northwestern Polytechnical University, 2006)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE

SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2015

Declaration

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

KANG Wei

25 February 2015

ACKNOWLEDGEMENTS

In memory of my mother for her everlasting love and encouragement.

This thesis would not have been accomplished without the love and help of many people. Therefore, I would like to express my gratitude to them here in this section.

First of all, I am greatly indebted to my supervisor, Prof. Anthony K. H. TUNG, for his precious guidance over the past few years. His creativity and devotion to research have impressed me a great deal, which encourages me all the time during my entire PhD study. Thank him for his discussion and inspiration so that I can always stay highly motivated in research. I am also grateful to him for inviting us to dinner or karaoke sessions from time to time, from which I have learnt the necessity and importance of balancing work with life.

I would like to thank Prof. Kian-Lee TAN and Prof. Roger ZIMMERMANN to be the members of my Thesis Advisory Committee (TAC). I am really grateful that they are always willing to attend the regular TAC meetings. Their invaluable suggestions have inspired and enlightened me on the formation of new ideas. I am also pretty grateful to my thesis examiners, i.e., Prof. Stéphane BRESSAN, Prof. Roger ZIMMERMANN and Prof. Guoliang LI, for their precious comments and suggestions, so that I can better refine and polish my thesis after the oral defense. In addition, I would like to convey my

sincere gratitude to the NUS Graduate School for Integrative Sciences and Engineering (NGS) for providing me with the generous scholarship.

My special gratitude goes to my friends in the Family of Christ church, especially Prof. Kian-Lee TAN, WANG Zhengkui, XIAO Qian, WANG Huiju, ZENG Yong, CAO Nannan and CHU Yan. Whenever I encounter any difficulties in either research or daily life, they are always there, ready to help me and encourage me. Particularly, Prof. Kian-Lee TAN, a respectable person with great humility, considerateness and wisdom, sets an example to me not only as a prominent professor but also as a spiritual mentor and sincere friend. Having these trustworthy friends is a big fortune and blessing to me, making me feel the warmth and comfort like at home.

I am pretty grateful to the colleagues and labmates in the database research labs for their kindly help. We have built a strong battle companion-like friendship by helping and getting along with each other over the past few years. They are ZHOU Jingbo, ZHENG Yuxin, TANG Ruiming, BAO Zhifeng, LI Lu, LI Hao, ZENG Zhong, ZHAO Feng, SHI Lei, WANG Guoping, WANG Fangda, FAN Qi, GUO Qi, LIU Qing, LI Meiying, LI Yuchen and so on. I would like to extend my gratitude to my collaborators in these years, including ZHAO Feng, LI Xinyu, SONG Qiyue, Shubham GOYAL and so on. I would also like to thank my friends LI Xinyang and DENG Linli for their encouragement, and thank YUAN Xintong and her family for their hospitality.

Lastly, my deepest love is reserved for my beloved family, i.e., my father, my mother and my brother, for their endless affections and support. Without their constant encouragement and confidence in me, I can never make any progress as such. I would like to ascribe all my progress and achievements to my family, and dedicate this thesis to my mother in memory of her and her everlasting affection for me.

CONTENTS

SUMMARY

The proliferation of social media services has led to the production of huge amounts of data, which raises great challenges to information acquisition, integration and digestion. To extract compact yet useful information, many algorithms have been proposed to summarize social media contents, e.g., tweets and news feeds. However, it remains challenging to extract summaries efficiently and support the interactive exploration of such data. Most existing methods also extract summaries without considering the semantic meanings and relationships in those summaries. Even with the extracted information, users may still find it hard to obtain knowledge in conformity with their preferences.

To tackle these challenges, we propose two novel summarization approaches in this thesis to generating hierarchical summaries. One approach generates summaries from spatiotemporal social media contents and builds a system to visualize the summaries in hierarchical tag clouds. The other approach focuses on introducing semantics into each summary. In addition, a system with four data analytics tools is built to manage social media contents and extracted knowledge via Wikipedia.

Specifically, we first propose Vesta which enables users to extract and interactively explore summaries of social media contents published in a certain spatiotemporal range.

These summaries are represented using a novel concept called hierarchical tag clouds, which allows users to zoom in/out to explore more specific/general tag summaries. A novel biclustering approach is proposed to extract summaries, from which topic hierarchies are generated for partitions of data. At runtime, topic hierarchies in certain partitions are merged to form tag hierarchies, which are used to construct hierarchical tag clouds for visualization.

Next, we propose Heron to generate hierarchical summaries from any set of social media contents. It makes use of the DBpedia ontology, through which semantically hierarchical relationships are introduced into each summary. Specifically, a summary consists of a set of semantically related Wikipedia entities which are extracted from social media contents. The entities are further classified into different subsets, which are mapped to the corresponding classes in a sub-hierarchy of the DBpedia ontology to reveal subsumptive relationships. We propose a model named multi-level Naive Bayes Classifiers to refine the classes of entities so as to reduce inaccuracies and inconsistency in Wikipedia. Considering the probability that many entities may be mapped to a single class, we further propose to select the top-ranked entities for each subset of a summary.

Finally, we present a novel system named Trendspedia, which brings proper context to continuously incoming social media contents, so that massive amounts of information can be indexed, organized and analyzed around Wikipedia entities. Four data analytics tools are employed. With this system, users can easily pinpoint valuable information and knowledge, and navigate to other closely related entities through an information network for further exploration.

Extensive experimental studies have verified the efficiency, effectiveness and scalability of our approaches. We believe that our summarization approaches, as well as the Trendspedia system, can greatly promote and facilitate the exploration of insights hidden in huge numbers of social media contents.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

As social media services have become ubiquitous nowadays, a huge number of social media contents, e.g., tweets on Twitter and news feeds on Facebook, are published every single day. For instance, Twitter is seeing 500 million tweets per day in 2015[1]. Such a huge volume of data brings great challenges in terms of information acquisition, integration and digestion. One may feel overwhelmed when facing mountains of data every day, and may also miss important information in the data flow if one follows a large number of people. In addition, while social media contents propagate and exchange information at an unprecedented speed, they are also characterized by noise and redundancy. In response, much research effort has been invested in the summarization of social media contents, with the aim of extracting useful and compact information while filtering out the trivial. Popular methods include topic discovery [84, 63, 53],

[1] https://about.twitter.com/company

event detection [19, 102, 18], time-awareness summarization [81, 76] and so on.

Among various types of social media contents, geo-coded ones are increasingly becoming more common. These contents include both locational and temporal information which greatly enriches the published posts. By exploring and summarizing these contents, we can discover what people are talking about in certain regions during a certain period of time. For example, in the 2012 US presidential election, Obama and Romney tried to win voters in key swing states by getting their campaign staffers to analyze newly published social media contents that were related to the election in each of these states, and then adjusted their campaign strategies according to the analysis. Therefore, summarizing social media contents with spatiotemporal information effectively and visualizing the resultant summaries wisely would definitely enhance the understanding of massive amounts of data.

In addition to generating summaries from geo-coded contents, we believe that introducing and exploring semantic meanings in summaries is also substantially important for information extraction and digestion. However, most existing research efforts mainly generate summaries for documents and social media data from statistical perspectives, thereby leaving the semantic meanings and relationships in summaries unexplored.

Furthermore, even with extracted information, users may still find it difficult to obtain personalized knowledge in conformity with their preferences. This happens because of the lack of an effective and systematic mechanism to organize the massive amount of information and extracted knowledge so that users can easily pinpoint what they are interested in.

In this thesis, we extract knowledge by analyzing social media contents and propose effective solutions to the above problems. Specifically, we first propose two summarization approaches. One approach is used to generate summaries from spatiotemporal

social media contents, and a system is implemented to visualize the summaries in hierarchical tag clouds. The other approach tries to extract hierarchical summaries by introducing semantic meanings and relationships into each summary. Besides the two approaches, we also present a system named Trendspedia, which is designed to manage massive amounts of information and extracted knowledge effectively by leveraging Wikipedia.

1.2 Research Problems and Challenges

Challenges in the summarization of social media contents mainly involve the efficiency and effectiveness of the approaches. To visualize the generated summaries and enable interactive exploration of them, we need to figure out how to organize the elements in each summary according to their meanings and importance. It is also challenging to introduce semantics to summaries when most existing efforts make use of statistical information. With an abundance of information and extracted knowledge, another problem is how we can effectively organize them and enable users to pinpoint the knowledge they are interested in with great ease. Next, we briefly introduce the problems tackled in this thesis, as well as the challenges that we encountered when trying to solve these problems.

1.2.1 Generating and Visualizing Summaries

The first problem that we address in this thesis is the extraction of summaries from spatiotemporal social media contents, so as to help users explore what people are talking about in certain regions during certain periods of time. To provide interactive exploration of the summaries, we propose a new concept of the hierarchical tag cloud and build a system named Vesta to visualize the summaries in hierarchical tag clouds. A

hierarchical tag cloud is a visualized form of a summary. Keywords in a summary are called tags in the hierarchical tag cloud, which organizes the tags at different levels of depth according to their degrees of generality. In this manner, users are enabled to explore and better understand the summaries by zooming in/out through the hierarchical tag clouds interactively.

One big challenge associated with providing interactive hierarchical tag clouds is the development of efficient methods to summarize social media contents. Latent Dirichlet allocation (LDA) [15] is a popular topic model for summarizing and extracting topics from documents, and various extensions [13, 12] have been studied, including extending LDA to deal with short documents like tweets [71]. However, LDA-based methods cannot handle huge amounts of data efficiently since they often perform inference by adopting MCMC algorithms such as Gibbs sampling [28]. The streaming nature of social media content data renders the summarization task even more challenging. Even an efficient summarization method might not be able to handle huge amounts of data easily when scaled up, and might not guarantee a relatively short response time.

Another challenge to providing hierarchical tag clouds is organizing the keywords in each summary at different levels to reflect different degrees of generality. This is important for users to gradually understand the meanings conveyed in a summary, level by level.

1.2.2 Generating Summaries with Semantic Meanings

The existing summarization methods greatly promote information integration and enable users to understand a large number of social media contents in a more concise and effective manner. However, most of these methods leave the exploration of semantic meanings and the relationships that summaries convey untouched, mainly because the summaries are often generated based on statistical information such as co-occurrence

and frequency of terms [15, 19, 84]. Therefore, the summaries generated by most existing methods fail to reveal insights hidden in social media contents for lack of semantic information integration.

To tackle the second problem in this thesis, we propose an approach named Heron to generate another type of summary, i.e., the hierarchical summary, in which semantic meanings can be introduced and explored. A hierarchical summary consists of a set of closely related Wikipedia entities which are extracted from social media contents. The entities are divided into a few subsets, each of which contains entities classified as the same class, by making use of the DBpedia ontology (a hierarchy of classes with subsumptive relationships). The subsets of entities in a summary are connected in a hierarchical structure, which corresponds to a sub-hierarchy of the DBpedia ontology to reveal the semantic relationships among the subsets.

To generate high-quality hierarchical summaries, it is important to map entities properly onto the DBpedia ontology based on the class labels of the entities. Although the class information of entities is available in Wikipedia, i.e., the infoboxes of Wikipedia entities, the crowdsourcing nature of Wikipedia inevitably leads to great inconsistency and inaccuracy (refer to Section 4.1.1 for details) [11]. To reduce the propagation of inconsistency and inaccuracy of class information from Wikipedia to summaries, it is better to refine the classes of entities before performing the summarization.

Many classification algorithms have been proposed in the literature. However, the characteristics of infoboxes in Wikipedia render traditional classification algorithms inefficient and impractical. These characteristics include high dimensionality (i.e., the number of properties altogether is quite huge), sparseness (i.e., the number of properties for an entity is often small, from a few to a dozen), and low degree of property overlap (i.e., different entities may have quite different sets of properties), all of which

present great challenges to the refinement of classes of Wikipedia entities.

Furthermore, many entities can be mapped to a single class node in the DBpedia ontology because they may belong to the same class. Thus it is also important to rank the entities properly according to some criteria, to provide users with the key entities.

1.2.3 Managing Information and Extracted Knowledge

Even with summaries generated to enable users to grip sketches instead of reading huge numbers of social media contents, users may still feel overwhelmed by the information in which they have no interest at all. Most people actually have their own preferences for knowledge within certain contexts, instead of the globally popular ones. Consider, for instance, the scenario where a tourist is going to visit a place that he has never been to before, or an investor plans to buy stocks of a certain company. It would be greatly beneficial if both of them had access to some well organized and continuously updated knowledge of how other people talk about their targets, and furthermore, if they could also obtain similar knowledge of a few more closely related entities via an information network. Such a demand for preference-based knowledge acquisition brings great challenges to traditional means of information retrieval and integration, such as search engines, whose users have to search for snippets of information that may interest them, and integrate and analyze the information by themselves.

To tackle these challenges, we build a collaborative Internet observatory platform named Trendspedia with the aim of bringing proper context to social media contents which are streaming in from the Internet. With Trendspedia, we try to index, organize, and analyze massive amounts of dynamic social media contents around Wikipedia entities, so that users can easily pinpoint useful information and analytical results by simply navigating to the Wikipedia entities they are interested in. To facilitate the exploration of relevant knowledge, users are empowered to navigate to closely related Wikipedia

entities effortlessly, through an information network.

1.3 Contributions

This section describes the contributions made in this thesis to the solving the above problems.

In the first part of this thesis, we propose a new concept named hierarchical tag cloud for the summarization and visualization of spatiotemporal social media contents. Our contributions are as follows:

- We propose a novel way to explore spatiotemporal social media contents via hierarchical tag clouds. Users are allowed to interactively drill down or roll up in the hierarchical tag clouds to understand the corresponding summaries at different levels of abstraction.

- We propose an efficient summarization approach by biclustering the social media contents based on formal concept analysis. We then generate and merge topic hierarchies to visualize the summaries in hierarchical tag clouds.

- To enhance the scalability, we further extend the summarization approach to a disk-based partition-and-merge scheme. At the partitioning stage, which is done offline, we split the spatiotemporal data space into partitions and generate summaries and the corresponding topic hierarchies for each partition. At runtime, topic hierarchies are merged to be visualized in hierarchical tag clouds.

- We implement all these mechanisms, and, based on them, build a semi-realtime system called Vesta.

In the second part of this thesis, we propose Heron, a new type of summarization

approach with the aim of introducing semantics into summaries. Our contributions in this respect are as follows:

- We generate hierarchical summaries, each of which not only contains semantically related entities but also has a hierarchical structure, corresponding to a sub-hierarchy of the DBpedia ontology, to reveal subsumptive relationships among subsets of entities.
- We propose a model named multi-level Naive Bayes Classifiers to refine the classes of entities before mapping them onto the DBpedia ontology, so as to reduce the propagation of inconsistency and inaccuracy in Wikipedia.
- Considering the possibility that many entities may be mapped to one single class in the DBpedia ontology, we introduce a ranking procedure to select the most relevant entities based on a score formula.
- The hierarchical summaries can also be easily visualized in hierarchical tag clouds.

In the third part of this thesis, we try to manage massive amounts of information and extracted knowledge effectively. Our contributions here are as follows:

- We build a system named Trendspedia to help users pinpoint information and knowledge easily, according to their preferences.
- We index, organize, and analyze dynamic social media contents effectively around Wikipedia entities.
- We implement four data analytics tools in Trendspedia and visualize the analytical results for better exploration and understanding.

In addition, two papers have been published in international conferences based on the first and third parts of this thesis [40, 39].

1.4 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, we review the related works. In Chapter 3, we propose to extract summaries from spatiotemporal social media contents and we build a system for interactive exploration of the summaries in hierarchical tag clouds. Chapter 4 introduces another summarization approach, so as to generate hierarchical summaries which convey semantic meanings and relationships. To effectively manage massive amounts of information and extracted knowledge, we further present a system named Trendspedia in Chapter 5, to index, manage, and analyze dynamic social media contents around Wikipedia entities. Finally, we conclude this thesis and discuss possible future directions in Chapter 6.

CHAPTER 2

RELATED WORKS

In this chapter, we review and synthesize related works. First, we study the different categories of summarization methods proposed in the literature. Then, we introduce some other relevant fields, including tag clouds for the visualization of summaries, multi-label classification and hierarchical classification, and knowledge discovery in social media.

2.1 Summarization

Various summarization methods have been proposed over the past decades with the aim of summarizing and extracting knowledge from large volumes of data. These methods were initially designed for information retrieval from textual records such as documents and news articles. Later, with the rapid development of many emerging scientific/commercial fields, such as biological technology and social media services, a large amount of data is produced every day, which thus demands efficient techniques

for the analysis of massive data. Summarization is consequently applied in these fields to extract key features and important points. Summarization methods can be divided into different categories according to the techniques adopted and objectives to achieve. Next, we review these methods in terms of categories in detail.

2.1.1 Topic Discovery

One popular category of the summarization approaches is the topic discovery, which attempts to extract different topics from textual records, such as a corpus of documents or a collection of news articles. A topic usually consists of a group of keywords or sentences describing one central fact or several closely related facts while keeping the redundancy minimized.

Many efforts try to select the most important sentences by assigning scores to sentences in documents. Barzilay and Elhadad [7] generated a summary for a text using a model of the topic progression derived from lexical chains, without requiring the full semantic interpretation of the text. They argued that lexical chains were a good indicator of the central topic of a text and presented a new algorithm to compute lexical chains in a text by merging multiple knowledge sources. With the lexical chains, they scored them so as to identify the strong ones, from which significant sentences were extracted. Gong and Liu [31] proposed two generic text summarization methods by selecting sentences that were highly ranked and different from each other, with the aim of achieving a wider coverage of a document's main content and less redundancy. In [25], an approach named LexRank was proposed to compute sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. Authors in [97] first assigned a score to each term in a document cluster and then picked up sentences that maximized the sum of the scores in the cluster. A Cluster-based Conditional Markov Random Walk Model and a Cluster-based HITS Model were proposed

in [89] to support multi-document summarization using the link relationships between sentences in a document set.

2.1.2 Topic Modeling

Some other researchers try to extract summaries through topic modeling approaches. Latent Dirichlet allocation (LDA) [15] is one popular generative probabilistic model for latent topic discovery in text collections. LDA is a three-level hierarchical Bayesian model, which models each document of a collection as a finite mixture over a set of latent topics. Each latent topic is characterized by a distribution over words. Since exact inference is intractable for LDA, the authors presented approximate inference techniques instead, based on variational methods and an EM algorithm for empirical Bayes parameter estimation. Hierarchical LDA (hLDA) [13, 12] is an extension of LDA to learn topic hierarchies through the Nested Chinese Restaurant Process. The authors proposed a nested Chinese restaurant process and showed how to use the process to do Bayesian nonparametric inference of topic hierarchies.

Inspired by LDA and hLDA, various other topic modeling methods have been proposed, such as the correspondence LDA models [14] and the topic-sentiment models [51]. Three hierarchical probabilistic mixture models were presented in [14] to model the joint distribution of both types and the conditional distribution of the annotation given the primary type in annotated data. To detect sentiment and topic simultaneously from text, Lin and He proposed a probabilistic modeling framework called joint sentiment/topic model based on LDA [51]. To solve the topic-driven reader comments summarization (TORCS) problem, Ma et al. in [53] introduced a Master-Slave Topic Model (MSTM) and an Extended Master-Slave Topic Model (EXTM) to discover and summarize topics in readers comments and the related news articles. Both models perceived a news article as a master document and each of its comments as a slave document, and

grouped comments into topic clusters. Two ranking mechanisms were adopted to select most representative comments from each comment cluster. In another work [22], the authors proposed a model called Uni-Topical Blockmodels to capture topics from tweet replies among groups of Twitter users. Unlike most LDA based methods which rely on the bag-of-words assumption, some authors went further by taking word order and dependency into consideration. Griffiths et al. presented the HMMLDA model in [32] by taking care of both short-range syntactic dependencies and long-range semantic dependencies between words. The authors in [90] proposed a topical n-gram model based on LDA, with the aim of discovering topics and topical phrases. The model can automatically determine unigram words and phrases according to context and then assign mixture of topics to individual words and n-gram phrases.

Since LDA-based methods often perform approximate inference by adopting MCMC algorithms such as Gibbs sampling, they are not able to handle huge amounts of social media data efficiently. In Chapter 3, we propose an efficient summarization approach by biclustering spatiotemporal social media contents. After the summaries are generated, we only adopt hLDA to generate tag hierarchies to help visualize the summaries. We show in the experimental study that hLDA cannot handle large amounts of content data directly for the analysis of topic hierarchies. Our work in Chapter 3 differs from hLDA mainly in the following aspects. (1) We discover summaries and the related contents from which the summaries are extracted simultaneously. (2) We generate discrete summaries which capture various interesting topics while hLDA generates topics at different levels of abstraction. (3) We generate a tag hierarchy for each summary while hLDA generates a topic hierarchy for a corpus of documents.

As the biclustering based summarization approach will be introduced in Chapter 3, next we introduce the background of biclustering.

2.1.3 Biclustering

Biclustering [37] was first proposed decades ago and became popular after Cheng et al. [20] adopted it for gene expression data analysis. Many approaches were proposed in the field of Bioinformatics to do biclustering in both gene and condition dimensions simultaneously to generate biclusters, each of which has a set of genes expressing coherently in a set of conditions [20, 94, 93]. It is also used in text analysis, referred to as "co-clustering", to analyze dyadic data [74, 83], e.g., the document and word co-occurrence frequencies. Some research efforts were spent in extending existing co-clustering to support constraints on both words and documents [83]. A two-dimensional contingency table can be perceived as an empirical joint probability distribution of two discrete random variables, which converts co-clustering to an optimization problem in information retrieval. The optimal co-clustering results in the maximum mutual information between the clustered random variables subject to constraints on the number of row and column clusters. With this idea, an information theoretic co-clustering framework was proposed in [23] to increase the preserved mutual information by interlacing the row and column clusterings iteratively.

However, finding the largest bicluster is NP-complete for almost all variants of the biclustering problem [20, 54]. As a result, most biclustering approaches are designed with heuristics in a non-deterministic manner, thereby often taking a long time to converge [54, 93]. Recently, some authors [30] applied biclustering to the analysis of social media data. They proposed to do biclustering by means of formal concept analysis to extract groups of users sharing similar interests and discover communities of users coming from similar groups. Formal concept analysis [27, 70], or FCA for short, is a data analysis method in growing popularity across various domains, which is widely used to discover relationships between a set of objects and a set of attributes. Although biclustering using FCA is more efficient than general biclustering approaches, this approach

is still unable to scale to huge amounts of social media data. Besides, it tends to generate sparse biclusters but miss the dense ones [30]. Our proposed biclustering method in Chapter 3 also makes use of FCA to summarize social media data, but can handle large dataset more efficiently and generate denser biclusters.

2.1.4 Event Detection

Event detection is different from most categories of summarization in that it aims at discovering a significant or large-scale activity that is unusual with regard to normal patterns of behavior [42]. Event detection has similar definitions in many other works such as the discovery of abnormal aggregates in data streams [101], or looking for something special that happens at some specific time and place [69]. In other words, the major task of event detection is to detect or summarize abnormal things that are considered as different from usual status.

Many early research efforts focused on detecting events from streams of news stories. Yang et al. [95] applied hierarchical and non-hierarchical document clustering algorithms to a corpus of stories and found the resultant cluster hierarchies highly informative for the detection of previously unidentified events. Allan et al. in [2] conducted event detection over a stream of broadcast news stories by adopting a single pass clustering algorithm and a thresholding model incorporating the properties of events.

As social media services prosper nowadays, people often post real-time microblogs to report significant or bursty events, including social events such as a political campaign, a gun shot, as well as natural events such as an earthquake. By monitoring and exploring social media streams, researchers are able to detect events and stories which are characterized by a set of descriptive, collocated keywords. Sayyadi et al. [80] built a network of keywords and made use of community detection methods to discover and describe events in social streams. In [79], Sakaki et al. treated Twitter users as sensors.

They monitored tweets and detected events by a classifier of tweets based on features such as the keywords, the number of words and the context. In addition to descriptive keywords, events are also characterized by temporal/spatial features. By defining events as an information flow between social actors on a certain topic during a certain time period, authors in [100] detected events by exploring social media contents from temporal and social dimensions. Another group of authors in [49] tried to detect the occurrence of local events with geo-coded microblogs.

Different from event detection aiming at discovering abnormal/abrupt events, some researchers work on general summarization solutions with temporal, spatial or evolutionary characteristics. For instance, Yan et al. in [92] proposed a framework named Evolutionary Timeline Summarization (ETS) so as to produce evolutionary timelines consisting of individual yet correlated summaries given a collection of time-stamped web documents.

2.1.5 Social Media Contents Summarization

Since this thesis focuses on analyzing social media data, we next give a brief review of the related works on the summarization of social media contents exclusively. Although the summarization of documents has been studied for many years, the summarization of social media contents remains a new research direction and begins to attract more and more research interests in recent years owing to the new characteristics in social media contents, such as the limited length of a microblog and the streaming nature of social media data. The summarization approaches for social media contents can also be divided into specific categories, such as topic discovery and event detection as discussed above.

TweetMotif [63] is an application for Twitter topic summarization based on techniques such as near-duplicate detection, language modeling and set cover heuristics.

TUT [84] is a statistical model for the detection of interpretable trends and topics in social media, where a topic is a cluster of frequently co-occurred words. McCallum et al. [56] introduced an Author-Recipient-Topic (ART) model, based on latent Dirichlet allocation and the Author-Topic model, to learn topic distributions according to the relationships between people. Besides topic generation, event detection from social media data is another popular form of summarization [102, 18]. Authors in [19] summarized tweets for highly structured and recurring events by learning the underlying hidden state representation of events via Hidden Markov Models. Twevent [50] was proposed as a segment-based event detection system for tweets, which can help users to understand the topics attracting a large number of common Twitter actors. Other efforts are made in time-aware summarization, such as producing timelines by summarizing dynamic, quickly arriving, and large-scale tweet streams [81], and modeling tweet propagation to generate time-aware tweets summaries based on users' history and collaborative social influences [76]. However, very few of the existing works take into consideration the exploration of semantic meanings and relationships in summaries, which will be discussed in Chapter 4 of this thesis.

2.2 Other Related Works

Apart from summarization, this thesis is also relevant to several other research areas, including the visualization of summaries, multi-label/hierarchical classification as well as knowledge discovery in social media. Next we briefly review each of them respectively.

2.2.1 Visualization with Tag Clouds

Tag clouds are visual presentations of a set of words, also called "tags", selected by some rationale, in which attributes of the text (e.g., size, color) are used to represent features (e.g., frequency) of the associated terms. Kaser and Lemire [41] proposed algorithms to improve the display of tag clouds and to achieve a general 2-dimensional layout by using nested tables. Sinclair et al. [82] conducted an investigation to figure out when participants preferred tag cloud to traditional search interface in information query. The authors in [78] provided an extensive evaluation of tag clouds and introduced some guidelines for tag cloud construction. Bielenberg et al. [10] presented their tag cloud in a circular layout where tags locating nearer the center are more important. Dubinko et al. [24] proposed to visualize the evolution of tags within the Flickr online image sharing community. PubCloud [46] was built to summarize the query results in the PubMed database of biomedical literature using tag clouds. A user study conducted by the authors pointed out that tag clouds could better present descriptive information than the standard result list. In Chapter 3, we combine tag clouds and topic hierarchies to visualize the summaries of spatiotemporal social media contents in hierarchical tag clouds, which is suitable for large-scale tag representation.

2.2.2 Multi-label/Hierarchical Classification

To generate high-quality summaries with semantics, in Chapter 4 we propose a model named multi-level Naive Bayes Classifiers so as to refine the classes of entities. The model is related to both the multi-label classification [85] and the hierarchical classification [38]. Most traditional classification algorithms fall into the multi-class classification, which try to classify each of the instances under examination as only one of the multiple classes. When it comes to the multi-label classification, each instance can be associated with multiple labels or classes. To support multi-label classification

with a large number of labels, authors in [9] proposed to select a small subset of class labels to approximate the original label space. A pruned sets method was presented in [73], which took into account correlations between labels by treating sets of labels as single labels. Hierarchical classification is a type of the multi-class classification, which organizes the multiple classes to be predicted into a class hierarchy [3, 38, 75, 48]. For instance, Kumar et al. [45] proposed a hierarchical method to solve the K-class problem by using $K-1$ binary classifiers. The binary classifiers were organized in a binary tree with K leaf nodes, each of which represented one of the K classes. Our model of multi-level Naive Bayes Classifiers proposed in Chapter 4 is a combination of the multi-label classification and the hierarchical classification, which is specially designed to refine the classes of entities based on the DBpedia ontology.

2.2.3 Knowledge Discovery in Social Media

Nowadays, we have entered an era where a huge amount of data is continuously produced every day. In addition to the efforts in developing summarization algorithms, many studies have been performed to gain various insights from massive social media data so as to move forward from data to information to knowledge.

Xiang and Gretzel [91] simulated the travel planning process of travelers to investigate the role of social media in online travel information search, and proposed suggestions for better online marketing strategies based on their findings. Yates et al. [96] studied social media as knowledge management systems for disaster and emergency management, including the influence of social media on knowledge sharing, reuse, and decision-making, and how knowledge can be effectively maintained in these systems. Authors in [6] attempted to analyze and interpret the context of Twitter users, including interests, intentions and activities, from the real-time data flow of Twitter messages. Bandari et al. [5] leveraged a multi-dimensional feature space extracted from articles to

predict the popularity of news items in social media.

Although the existing efforts successfully derive various knowledge from social media data, it is more desirable to provide users with effective means to locate knowledge easily according to their preferences, considering the overwhelming data produced and huge amounts of knowledge derived every day. In Chapter 5, we introduce a novel system which organizes social media contents and the extracted knowledge around Wikipedia entities, so as to help users pinpoint the information and knowledge in which they are interested.

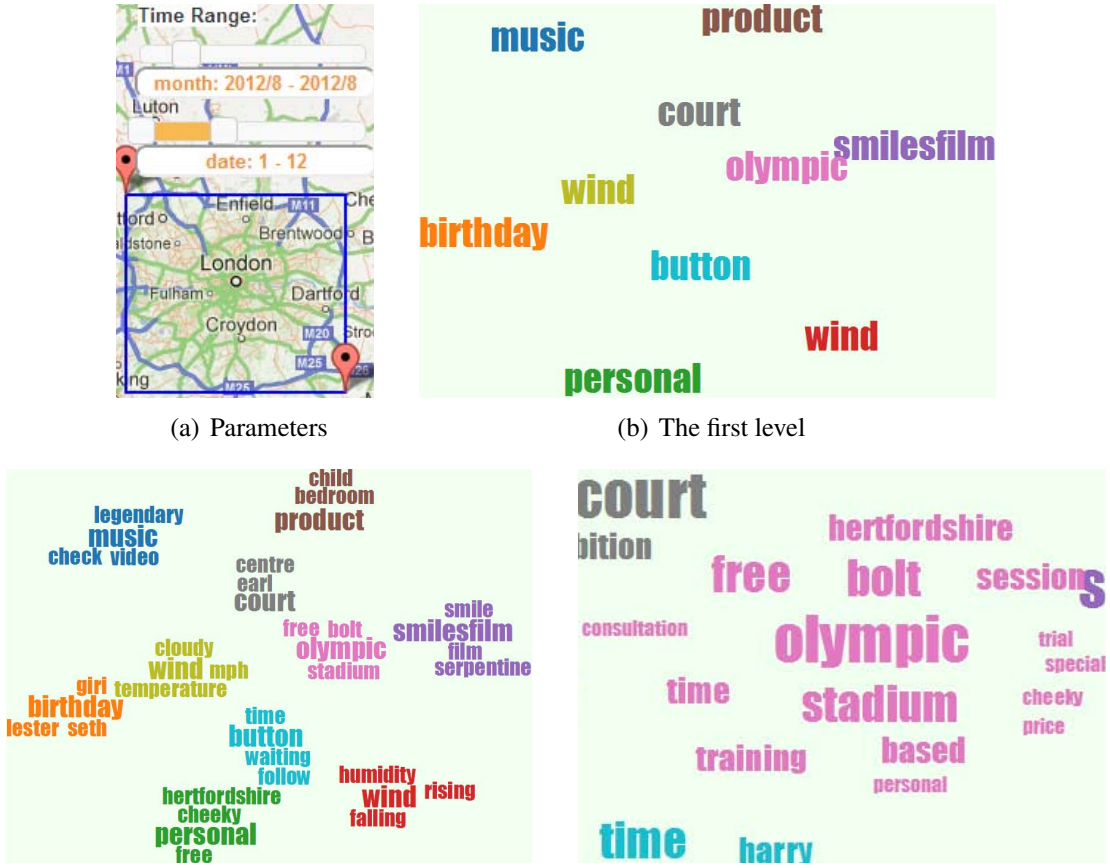
CHAPTER 3

INTERACTIVE HIERARCHICAL TAG CLOUDS FOR SUMMARIZING SPATIOTEMPORAL SOCIAL MEDIA CONTENTS

3.1 Overview

In this chapter, we propose a system called Vesta[1] that enables users to interactively browse the summaries of social media contents of user-specified spatiotemporal regions. To represent the summaries, we introduce the novel concept of the hierarchical tag cloud, which organizes tags (keywords) of a summary at different levels of depth based on their degrees of generality. In other words, tags at higher/lower levels have more

[1] Vesta stands for *v*isual *e*xploration of *s*ocial media contents via *ta*g clouds.



(a) Parameters (b) The first level

(c) The second level (d) The fourth level

Figure 3.1: Hierarchical tag clouds

general/specific meanings. Users can obtain a general idea of what is popular in a specified spatiotemporal region at first glance, and then zoom in to lower levels if they want to see more details. Therefore, the increasing depths of a hierarchical tag cloud can effectively organize tags of a summary at different levels of abstraction and present details to users step by step.

As an example, Figure 3.1 illustrates how users can interactively explore what were happening in London during the 2012 Olympic Games via hierarchical tag clouds. By setting a time range and selecting a geographic region as in Figure 3.1(a), the top 10 most interesting hierarchical tag clouds are displayed, where different colors represent

different topics. Initially, only tags at the first level of each hierarchical tag cloud are displayed (Figure 3.1(b)), with the largest font size conveying the most general meanings. When zoomed in, tags at subsequent levels of each tag cloud are gradually displayed in increasingly smaller font sizes around the first level tags to provide more specific meanings (Figures 3.1(c) and 3.1(d)). This process of hierarchical browsing of tag clouds can be repeated to trigger the display of the tags at any appropriate level.

As shown in Figure 3.1(b), one tag at the first level is "olympic", which summarizes the pink tag cloud and indicates that the tag cloud talks about the Olympic Games. To explore the tag cloud more, we zoom into the second level, as in Figure 3.1(c). We can see that more tags are displayed, including "bolt", "stadium" and so on. Figure 3.1(d) shows the pink tag cloud when we zoom into the fourth level, with even more tags added around "olympic", including "training" and "time" at the third level, and "price" and "trial" at the fourth level.

Exploring social media contents interactively in hierarchical tag clouds is a novel and useful operation. It assists users in exploring different topics by enabling them to only view summaries instead of having to read plenty of contents directly. Users can drill down or roll up in the tag clouds to better understand the discovered knowledge interactively and hierarchically. They can also click any tag in a tag cloud to see the related contents if they want to know the exact underlying context. Moreover, users are allowed to specify two sets of spatiotemporal ranges to compare summaries of different regions, for which common tags are highlighted in *italic* type.

To make interactive hierarchical tag clouds possible, one big challenge is to develop efficient methods to summarize social media contents. In this chapter, we propose an efficient biclustering approach based on formal concept analysis [27, 70]. The results are called biclusters, and each consists of a set of tags (keywords in contents) and a set of contents, with the tags frequently co-occurring in the contents. The contents are

clustered together due to the common tags they share, which means that they are quite likely to discuss similar things. As such, tags in a bicluster serve as a summary of the contents. Thus the summarization of social media contents is converted to the generation of biclusters. In addition, biclustering clusters tags and contents simultaneously, so that each bicluster contains both tags as a summary and contents from which the summary is extracted, with the contents supplementing the tags to provide more context for a summary. This distinguishes our approach from normal clustering that often clusters in a single direction, and from LDA-based methods that generate only summaries. Although finding the largest bicluster is NP-complete [20, 54] and many heuristic biclustering methods converge slowly, our approach makes use of formal concept analysis to generate "full-density" biclusters (i.e., each tag of a bicluster appears in each content of that bicluster) very efficiently. "Full-density" biclusters can be strict, but it is easy to relax a bicluster by adding more tags/contents or merging similar biclusters.

To further enhance the system scalability and visualize the summaries in hierarchical tag clouds, we propose an efficient two-phase, disk-based partition-and-merge scheme in Vesta. The partitioning phase, which is done offline, consists of three steps. In the first step, we split the spatiotemporal social media data into partitions. For instance, each partition contains one day's contents for a spatial region in our case. In the second step, we summarize the social media contents using the proposed biclustering approach. In the third step, we apply the hierarchical LDA (hLDA) model [13, 12] to generate for each partition a topic hierarchy. Topic hierarchies will be merged in the subsequent merging phase to form tag hierarchies, which organizes tags of merged biclusters at different levels, from general to specific. In this way, users can visualize the summaries in a hierarchical fashion. At the end of the partitioning phase, we have for each partition a set of biclusters and a topic hierarchy generated by using the contents in these biclusters. The partitioning phase can also be easily carried out in parallel.

The merging phase is done at runtime when users query Vesta by specifying the spatiotemporal ranges. Vesta first computes the partitions that are covered by the query range and then proceeds to merge similar biclusters from the selected partitions. A probabilistic merging algorithm is proposed to combine the corresponding topic hierarchies to form tag hierarchies for visualization purposes. Vesta is efficient as most of the computationally intensive tasks are done offline in the partitioning phase; the runtime merging phase is fast, making Vesta an effective interactive visualization tool.

To select the most interesting summaries, we propose a score function customizable according to users' preferences. We also evaluate the mismatch problem quantitatively, so as to provide feedback to adjust the partition size adaptively.

To sum up, we make the following contributions in this chapter: (1) We propose a novel way to explore spatiotemporal social media contents via hierarchical tag clouds. (2) We propose an efficient summarization approach by biclustering the contents, and further extend it to a disk-based partition-and-merge scheme for better scalability. (3) We generate and merge topic hierarchies so as to visualize summaries in hierarchical tag clouds. (4) We implement all these mechanisms in a system called Vesta.

The rest of the chapter is organized as follows. We introduce some preliminaries and give the problem formulation in Section 3.2. Section 3.3 discusses our new biclustering method. In Section 3.4, we propose a partition-and-merge scheme, followed by the introduction of the system implementation in Section 3.5. Section 3.6 presents the experimental study, and Section 3.7 summarizes this chapter.

3.2 Problem Formulation

In this section, we first provide some preliminaries and then give the problem definition.

3.2.1 Preliminaries

Definition 3.1. ***Social media contents*** *(or* ***contents*** *for short) are textual microblogs, such as tweets, published by users in social media services. Each content is denoted by $c_i \in \hat{C}$, where $\hat{C}$ is the collection of all contents.* ***Tags*** *are meaningful keywords in contents after stop words removal. Each tag is denoted by $t_i \in \hat{T}$, where $\hat{T}$ is the collection of tags. A* ***social media matrix*** *$M_{\hat{T}\hat{C}}$ is a $|\hat{T}|$ by $|\hat{C}|$ matrix whose rows represent tags and whose columns represent social media contents.*

Definition 3.2. *A* ***bicluster****, denoted by (T, C), is a pair consisting of a subset T of tags $\hat{T}$ and a subset C of social media contents $\hat{C}$. The process of finding biclusters is called* ***biclustering****. A bicluster corresponds to a submatrix M_{TC} of the social media matrix $M_{\hat{T}\hat{C}}$, where the tag set T of the bicluster corresponds to the row set of the submatrix and the content set C to the column set.*

The value of any element M_{ij} in $M_{\hat{T}\hat{C}}$ can be 1 or 0, indicating whether tag t_i appears in content c_j or not. Therefore, biclustering in social media data analysis is the process to explore the social media matrix $M_{\hat{T}\hat{C}}$ to discover biclusters (submatrices) satisfying certain criteria. Density is commonly used as an effective measure for determining the quality of biclusters. Without loss of generality, we use density as one of the major measures to assess the quality of a bicluster in this chapter. Users can easily replace it with other measures such as variance [37] and mean squared residue [20].

Definition 3.3. *The* ***density*** *of (T, C), denoted by $den(T, C)$, is the non-zero rate of its corresponding submatrix M_{TC}, which equals the ratio of the number of 1's to $|T||C|$. The* ***size*** *of (T, C), denoted by $sz(T, C)$, is defined to be $\min(|T|, |C|)$.*

In a bicluster, a set of tags co-occur in a set of contents such that the tags can be viewed as a summary of the contents. The denser the bicluster, the more "consistent" the contents in a summary. A density threshold δ_{den} and size threshold δ_{sz} can filter out

sparse biclusters with very few tags or contents, thereby avoiding bicluster explosion. Having the biclusters, we visualize them in hierarchical tag clouds, which calls for the generation of tag hierarchy using the tags in each bicluster.

Definition 3.4. *If a tag set T can be divided into a few non-empty subsets T_1,T_2,...,T_n such that $T_i \cap T_j = \emptyset$, $\cup_{i=1}^{n} T_i = T$ and $T_i < T_j$ (i.e., every two different subsets follow a total order) for any $i, j \in \{1, 2, ...n\}(i < j)$, we say that T_1,T_2,...,T_n form a **tag hierarchy** $\mathcal{H}$ for the tag set T and that T_i contains tags at the ith level of the hierarchy.*

A tag hierarchy organizes the tags of a bicluster at different levels from general to specific. A hierarchical tag cloud is actually the visualized form of a tag hierarchy while a tag hierarchy defines the levels of tags for a hierarchical tag cloud. By showing the tags level by level in a hierarchical tag cloud, users can better understand the meaning of the tags and the relationships among them in an interactive way.

3.2.2 Problem Definition

By highlighting a geographic region R_{geo}, our aim is to (1) generate the top-k most interesting biclusters (T_i, C_i) $(1 \leq i \leq k)$, where $den(T_i, C_i) \geq \delta_{den}$ and $sz(T_i, C_i) \geq \delta_{sz}$, to summarize the set of social media contents $\hat{C}$ published within R_{geo} during a user-specified period of time R_{tim}, and (2) build a tag hierarchy $\mathcal{H}_i$ for each tag set T_i so as to visualize these summaries in hierarchical tag clouds. The interestingness of the biclusters is measured by a score function, which will be introduced in Section 3.4.4.

With hierarchical tag clouds, users are empowered to explore any geographic region of interest, or even discover commonalities and uniqueness by comparing different regions.

3.3 Biclustering Approach

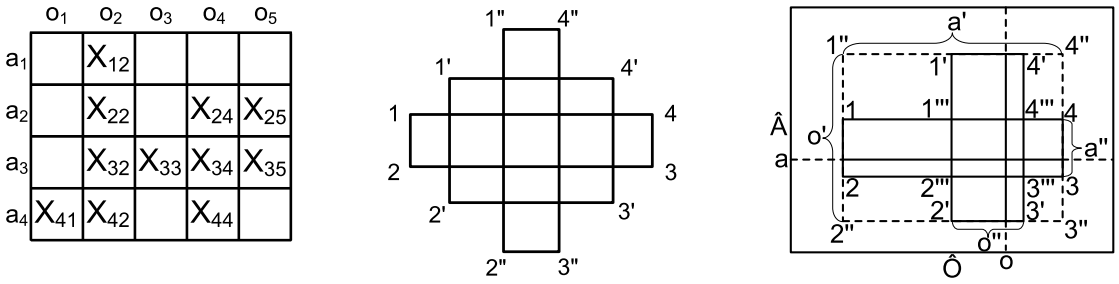
Most biclustering algorithms iterate many times prior to convergence, which renders the execution time large and unpredictable. Besides, biclustering was initially proposed to analyze gene expression data often in small size with no or a small number of empty values. Social media content data, however, is usually large and sparse, further disqualifying the application of common biclustering algorithms in social media data analysis. Next, we propose an efficient and deterministic way of finding biclusters based on the formal concept analysis (FCA).

3.3.1 Introduction to Formal Concept Analysis

Definition 3.5. *A **formal context** is a triplet $(\hat{A}, \hat{O}, I)$ where $\hat{O}$ is an object set, $\hat{A}$ is an attribute set and $I \subseteq \hat{A} \times \hat{O}$ represents the relationships of objects in $\hat{O}$ and attributes in $\hat{A}$. A pair (A, O), where $A \subseteq \hat{A}, O \subseteq \hat{O}$, is called a **formal concept** of the formal context $(\hat{A}, \hat{O}, I)$ if it satisfies the* fullness *property: $\forall_{a \in A, o \in O}(a, o) \in I$, and the* maximum *property: $\forall_{o \notin O} \exists_{a \in A}(a, o) \notin I$ and $\forall_{a \notin A} \exists_{o \in O}(a, o) \notin I$.*

Every formal concept (A, O) is *full* and *maximal*: being full means that every object in O has all attributes in A, while being maximal means that, if any attribute $a \notin A$ (or any object $o \notin O$) is added to A (or O), $(A \cup \{a\}, O)$ (or $(A, O \cup \{o\})$) is not full and thus not a formal concept any more.

Example 3.1. *Figure 3.2(a) shows a formal context $(\hat{A}, \hat{O}, I)$ where $\hat{A} = \{a_1, a_2, a_3, a_4\}$, $\hat{O} = \{o_1, o_2, o_3, o_4, o_5\}$. The cells with X_{ij} means a_i is an attribute of object o_j, i.e., $(a_i, o_j) \in I$. (A, O) is a formal concept where $A = \{a_2, a_3, a_4\}, O = \{o_2, o_4\}$. (A, O) is full because every object $o_j \in O$ has all the attributes in A. It is also maximal. If we add in any attribute or object which is not in A or O, o_5 for instance, (A, O) would not be a formal concept because $(a_4, o_5) \notin I$.*



(a) Example (b) Change in shape (c) Four types

Figure 3.2: Formal concepts

Compared with the definition of social media matrix, it is obvious that a formal context can also be perceived as a matrix M whose rows are attributes and whose columns are objects. The value of an element M_{ij} is set to 1 if the *j*th object has the *i*th attribute, 0 otherwise. Similarly, a formal concept (A, O) can be perceived as a "full-density" bicluster whose density equals 1. From this perspective, (A, O) being full means the submatrix of the bicluster has no empty value while (A, O) being maximal means the submatrix will have empty value(s) if any $a \notin A$ or $o \notin O$ is added to A or O. Therefore, the problem of generating biclusters now becomes the generation of formal concepts.

3.3.2 Properties of Formal Concept

A partial order "$\leq$" can be defined in a formal context: given two formal concepts (A_1, O_1) and (A_2, O_2), we have $(A_1, O_1) \leq (A_2, O_2)$ if $O_1 \subseteq O_2$ (or $A_1 \supseteq A_2$ equivalently). This also implies a relationship between the object set and attribute set of a formal concept in terms of set size. That is, the larger the size of the object set, the smaller that of the attribute set, vice versa. This actually follows, according to Galois theory, the antitone Galois connection [29] which is defined as a pair of antitone functions $F : \mathbb{A} \to \mathbb{B}$ and $G : \mathbb{B} \to \mathbb{A}$ between two partially ordered sets $\mathbb{A}$ and $\mathbb{B}$, such that $\beta \leq F(\alpha)$ iff $\alpha \leq G(\beta)$ where $\alpha \in \mathbb{A}, \beta \in \mathbb{B}$. Here F can be viewed as a function

mapping an attribute set to an object set while G as a function mapping an object set to an attribute set. Next we introduce the Galois operators "$'$" and "$''$" which work as the above functions.

Given an attribute set $A \subseteq \hat{A}$ and an object set $O \subseteq \hat{O}$,

$$A' = \{o \in \hat{O} | \forall_{a \in A}(o, a) \in I\}$$
$$O' = \{a \in \hat{A} | \forall_{o \in O}(o, a) \in I\} \tag{3.1}$$

where A' is an object set, every object in which has all attributes in A, and O' is an attribute set, every attribute in which is an attribute of all objects in O. Similarly, A'' is an attribute set by applying the Galois operator twice to A (or once to A') and O'' is an object set by applying the Galois operator twice to O (or once to O'). The operator "$''$" is monotone (i.e., $A'' \subseteq B''$ if $A \subseteq B$), idempotent (i.e., $(A'')'' = A''$), and extensive (i.e., $A \subseteq A''$) [30].

Lemma 3.1. *Given any attribute set $A \subseteq \hat{A}$ and any object set $O \subseteq \hat{O}$, both (A'', A') and (O', O'') are formal concepts.*

Proof. We only prove that (A'', A') is a formal concept, (O', O'') can be proven similarly. According to Eq. 3.1, it is easy to note that $A'' \subseteq A$, $A' \subseteq O$ and $\forall_{a \in A'', o \in A'}(a, o) \in I$. Next we prove (A'', A') is maximal by contradiction. Suppose the object set A' is not maximal. There must exist an object $o' \notin A'$ that has all attributes in A''. Since $A \subseteq A''$ ("$''$" is extensive), o' has all attributes in A. This contradicts Eq. 3.1 stating that A' contains all objects having every attribute in A. Suppose the attribute set A'' is not maximal. There must exist an attribute $a' \notin A''$ shared by all objects in A'. This contradicts $A'' = \{a \in \hat{A} | \forall_{o \in A'}(o, a) \in I\}$ according to Eq. 3.1, meaning that all attributes shared by every object in A' are included in A''. Hence the proof. □

Example 3.2. *Consider the formal context in Figure 3.2(a). Given $A = \{a_2, a_3\}$, we*

have $A' = \{o_2, o_4, o_5\}$*, where each object in* A' *has all attributes in* A*, and* $A'' = \{a_2, a_3\}$*, where each attribute in* A'' *is an attribute of any object in* A'*, by applying the Galois operators. It is easy to know that* (A'', A') *is a formal concept according to its definition.*

This lemma provides a way to generate formal concepts. Given any attribute or object set, we can apply the Galois operator once and twice respectively to generate the attribute and object sets of a new formal concept. However, should we enumerate all the possible attribute or object sets to generate formal concepts (which leads to $2^{|\hat{A}|} + 2^{|\hat{O}|}$ different sets for a formal context $(\hat{A}, \hat{O}, I)$ in the worst case)? Besides, the formal concepts in a formal context can be ordered to form a concept lattice according to the inclusion relationships of the object or attribute sets. The number of concepts in the lattice is also up to $2^{\min(|\hat{A}|,|\hat{O}|)}$ in the worst case.

To answer the above question, we need to investigate how the size of an attribute or object set affects the size of its resultant formal concept. It is illustrated in Figure 3.2(b) where formal concepts are represented by rectangles with the length representing the size of the object set and height representing that of the attribute set. Given three attribute sets A_1, A_2 and A_3 ($A_1 \subseteq A_2 \subseteq A_3$), their formal concepts are (A_1'', A_1'), (A_2'', A_2') and (A_3'', A_3') which are denoted by rectangle 1234, $1'2'3'4'$ and $1''2''3''4''$ respectively in Figure 3.2(b). Note that $A_1' \supseteq A_2' \supseteq A_3'$ because a smaller attribute set is likely to be shared by a larger object set and vice versa. Similarly, we have $A_1'' \subseteq A_2'' \subseteq A_3''$. Thus, if we expand A_1 to A_2 and even to A_3, the object set A_1' in the corresponding formal concept starts shrinking and the attribute set A_1'' starts expanding. This can be reflected in Figure 3.2(b) by changing the shape of the formal concept from rectangle 1234 to $1'2'3'4'$, and to $1''2''3''4''$. Likewise, given any object set O, when it expands by adding more objects in it, the process can be reflected by changing the shape of the formal concept (O', O'') from $1''2''3''4''$ to $1'2'3'4'$ to 1234. The above dis-

cussion indicates that in a formal concept we can expand the attribute (or object) set by shrinking the object (or attribute) set. In this chapter, we generate formal concepts based on each attribute and object, which greatly reduce the number of formal concepts to be generated to at most $|\hat{A}| + |\hat{O}|$. Formal concepts with larger attribute (or object) sets can be generated easily by shrinking their object (or attribute) sets.

3.3.3 Generating Biclusters

Given any attribute $a \in \hat{A}$ and any object $o \in \hat{O}$, we can find their corresponding formal concepts (a'', a') (we write $(\{a\}'', \{a\}')$ as (a'', a') for simplicity) and (o', o''), which are shown in Figure 3.2(c) as rectangle 1234 and $1'2'3'4'$ respectively. They can be viewed as "full-density" biclusters according to the fullness and maximum properties. Besides, there are two other rectangles $1'''2'''3'''4'''$ and $1''2''3''4''$. The former is the overlap of 1234 and $1'2'3'4'$, which is too tight to be used as a bicluster since it often leads to very small attribute set and object set. The latter is treated as extended formal concept (by allowing the density less than 1) and used to generate biclusters in [30]. We find this form also less qualified since it often includes many empty values thus greatly decreases the density. The authors in [30] set a threshold to filter out those with small density. However, this causes a problem that when they filter out some less dense extended formal concept $1''2''3''4''$, they also discard the "full-density" formal concepts 1234 and $1'2'3'4'$ that are within $1''2''3''4''$.

Therefore, we use the two real formal concepts 1234 and $1'2'3'4'$ to generate biclusters in this chapter. They differ from each other in terms of shapes. Specifically, 1234 tends to have a larger object set and smaller attribute set while $1'2'3'4'$ tends to have a smaller object set and large attribute set. In terms of bicluster, 1234 corresponds to biclusters with more contents and fewer tags while $1'2'3'4'$ to those with fewer contents and more tags. Users can generate different forms according to their requirements. We

refer to the methods generating biclusters in the form of 1234 and $1'2'3'4'$ as *ours_tag* and *ours_content* respectively below. Also note that many duplicate biclusters could be generated because contents in a bicluster often have similar tags and each of the tag-content pairs would be used to generate biclusters. To avoid duplication, we only generate biclusters without overlap. That is, if a tag or content appears in the tag set or content set of a bicluster, it will not be used to generate other biclusters.

3.3.4 Relaxation

Since the "full-density" biclusters are too strict, we can relax them through either transformation or merging.

Transformation refers to adding tags or contents to a bicluster. We take the addition of tags as an example. Given a bicluster (T, C), the contents in C may have another set T' of tags which are not included in T because having those tags makes the bicluster no longer satisfy the fullness property. We now break the property to enlarge T by adding tags in T' to T. Tags in T' should be ordered so that each time the tag leading to the least density decrease is added to T. This continues until the next tag to be added makes the density of the bicluster less than a density threshold δ_{den}. To add even more tags, we can delete a few contents which lead to the largest density increase after the deletion so as to increase the bicluster density first.

Merging can also result in relaxed biclusters. In this chapter, we merge biclusters sharing common tags since those biclusters are more likely to discuss similar topics and may thus be merged. Readers can refer to Section 3.4.3 for details.

3.4 Partition-and-Merge Scheme

To further improve the scalability and reduce the system response time, we next extend the proposed biclustering approach to a disk-based partition-and-merge (PM for short) scheme. In the offline partitioning phase, we split the data space (all the content data) into partitions and generate biclusters for each partition. Then biclusters in certain partitions are merged efficiently at runtime given user-specified spatiotemporal parameters. To visualize the merged biclusters as hierarchical tag clouds, we adopt hLDA to produce topic hierarchies for partitions in the partitioning phase and generate tag hierarchies by merging the topic hierarchies at runtime.

3.4.1 Offline Partitioning

Three dimensions, i.e., longitude, latitude and time, need to be considered for partitioning. We first slice the data on a daily basis and then split the geographic space for each day adaptively according to data density by using a space-partitioning data structure such as kd-tree [8] or quadtree [26]. Note that the partitioning layout may be different for each day since the data distribution varies every day. Also note that slicing data on a daily basis is a tradeoff between two possible mismatches. Since the temporal parameter can be set to any consecutive days, covering more than one day in a partition may lead to a temporal mismatch between the parameter and partitions. On the other hand, covering less than 24 hours may lead to larger spatial partitions and increase the geographic mismatch discussed in Section 3.4.5.

3.4.2 Offline Pre-computation

Pre-computation of Biclusters

Once the data space is split, we can generate biclusters for each partition using the method proposed in Section 3.3. One potential problem is that partition-based biclustering may leave out biclusters that can only be formed by using contents of multiple partitions. This may arise if the number of contents regarding certain topics are small. In this case generating biclusters in as small size as possible may relieve the problem. However, this could produce uninteresting biclusters and lead to bicluster explosion.

Given this observation, we make an **assumption** that globally interesting summaries are also interesting in certain partitions. Specifically, if a bicluster about an interesting summary is generated from contents in some geographic region, biclusters about the similar summary can also be generated from contents in certain partitions of that region. We will introduce a score function to measure the interestingness of a summary later and validate this assumption in the experimental study.

Pre-computation of Topic Hierarchies

hLDA is extended from LDA to generate topic hierarchies for documents. We apply hLDA to social media contents in our context to generate topic hierarchies for different partitions. Topic hierarchies are generated after the biclustering process for each partition by using the contents of the biclusters. Topic hierarchies generated by hLDA are trees, with a few closely related tags in each node [12]. The tags at higher level nodes are more general and those at lower level nodes more specific, which provides the possibility of generating tag hierarchies with different levels of tags by merging topic hierarchies.

The pre-computation of biclusters and topic hierarchies can be done easily for dif-

ferent partitions in parallel.

3.4.3 Online Merging

Merging Biclusters

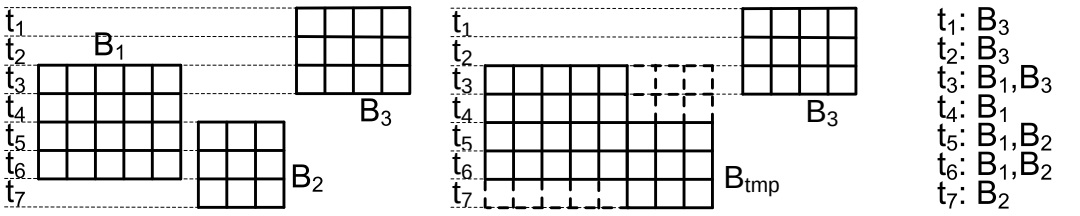
Given the spatiotemporal parameters, i.e., a time range R_{tim} measured in days and a geographic region R_{geo} measured in coordinates, biclusters in partitions falling within R_{tim} and R_{geo} need to be merged together to produce "unified" results. Note that if two biclusters (T_1, C_1) and (T_2, C_2) are merged to form a new bicluster (T, C), it will have all tags and contents from (T_1, C_1) and (T_2, C_2), i.e., $T = T_1 \cup T_2$, $C = C_1 \cup C_2$. Suppose the corresponding matrix of (T, C) is denoted by M_{TC}. The density of (T, C), $den(T, C)$, is thus the non-zero rate of M_{TC}. Next we give algorithm 1 for merging biclusters sharing common tags.

Algorithm 1: The Bicluster Merging Algorithm

Input: a group $\mathcal{P}$ of biclusters $(T_i, C_i)(i = 1, 2, ..., n)$ (use B_i to denote each bicluster for short), a user-defined density threshold δ_{den}
Output: a merged bicluster set $\mathcal{B}$

```
1  begin
2      let B_used = ∅;
3      foreach B_i ∈ P and B_i ∉ B_used do
4          let B_used = B_used ∪ {B_i};
5          find a subset P' of biclusters in P but not in B_used such that
           ∀_{B∈P'} B ⋂ B_i ≠ ∅;
6          sort B ∈ P' by |B ⋂ B_i| in descending order;
7          let B_new = B_i;
8          foreach B ∈ ordered P' do
9              merge B_new and B to form a new bicluster B_tmp;
10             if den(B_tmp) ≥ δ_den then
11                 let B_new = B_tmp;
12                 let B_used = B_used ∪ {B};
13         add B_new to B;
14 end
```

Given a group $\mathcal{P}$ of the biclusters and a density threshold δ_{den}, the algorithm produces a set $\mathcal{B}$ of merged biclusters. We initialize a set B_{used} storing used biclusters to empty in line 2 and start to merge biclusters by checking each bicluster in $\mathcal{P}$ but not in B_{used} in line 3. When we start from B_i we first label it as used in line 4. Line 5 and 6 try to find a set $\mathcal{P}'$ of biclusters having common tags with B_i and sort them according to the number of common tags in descending order so that those with more common tags can be merged with B_i earlier. To accelerate the search for $\mathcal{P}'$, we build an inverted list to map each tag to biclusters having that tag. Lines 7 to 13 merge B_i with the ordered biclusters in $\mathcal{P}'$. At first, B_i is merged with the first bicluster in $\mathcal{P}'$ to form a new bicluster B_{new}. Then B_{new} is merged with subsequent biclusters in $\mathcal{P}'$ in turn. Note that the merging action only happens if the density of the new merged bicluster is no less than δ_{den} (line 10) and that biclusters used to form B_{new} are labeled as used (line 12). After the merging phase finishes, B_{new} is added to $\mathcal{B}$ (line 13). The algorithm continues until all biclusters in $\mathcal{P}$ are used. We use an example below to illustrate the algorithm.



(a) Biclusters (b) Merging (c) Inverted list

Figure 3.3: Bicluster merging example

Example 3.3. *Consider the three biclusters B_1, B_2 and B_3 only sharing common tags (rows) in Figure 3.3(a). Suppose δ_{den} is 0.5. Algorithm 1 starts the merging from B_1 by searching for biclusters having common tags with B_1 as candidates which is B_2 and B_3. B_2 and B_3 should be sorted so that the one (B_2 here) with more common tags is merged with B_1 first. To search more efficiently, we build an inverted list as in Figure 3.3(c) showing which tag appears in which biclusters. Since B_1 have four tags, we can look up*

them in the inverted list to generate an ordered candidate set $\{B_2 : 2, B_3 : 1\}$ *(note that* B_1 *is removed from the list) easily. Each bicluster in the set has a number which is the frequency of the bicluster mapped to the four tags. This number also means the number of common tags shared with* B_1 *and thus can be used to sort the candidate biclusters.* B_2 *having more common tags is used to merge with* B_1 *first to form a new bicluster* B_{tmp} *whose tag (or content) set is the union of the tag (or content) sets of* B_1 *and* B_2*, as is shown in Figure 3.3(b). The density of* B_{tmp} *is* $(4\times5+3\times3)/(5\times8) = 0.725 > \delta_{den}$*, meaning the merging of* B_1 *and* B_2 *is acceptable. Next we use* B_3 *to merge with* B_{tmp}*. The new bicluster's density is* $(5\times8\times0.725+3\times4)/(7\times12) = 0.488 < \delta_{den}$*, meaning that we should cancel merging* B_{tmp} *with* B_3*. Since no further merging can be done, the algorithm outputs* B_{tmp} *and* B_3 *and terminates. In our system, we precalculate the density by assuming two biclusters were merged. If the density is less than* δ_{den}*, we do not merge them actually.*

Merging Topic Hierarchies

When biclusters are merged together, the topic hierarchies for partitions having those biclusters also need to be merged to form a tag hierarchy so as to visualize the new bicluster. Given that the same tag can appear at multiple levels of the topic hierarchies generated by hLDA, we next propose a probabilistic approach to generating "unified" tag hierarchies in which each tag only appears at one level. Each of the resultant tag hierarchies has one node containing several tags at each level, and each node has at most one child in terms of the tree structure.

Given m topic hierarchies T^i_1,T^i_2,...,$T^i_{n_i}(1 \le i \le m)$ to be merged together, where T^i_j denotes the tag set at the jth level of the ith topic hierarchy which has n_i levels in total, we first construct a vote-based tag-level matrix shown in Table 3.1. Rows in the matrix corresponds to tags while columns to levels. c_{ij} $(1 \le i \le u, 1 \le j \le v)$ in the

Table 3.1: Tag-level matrix

	$level_1$	$level_2$	...	$level_v$
tag_1	c_{11}	c_{12}	...	c_{1v}
tag_2	c_{21}	c_{22}	...	c_{2v}
...	...	...	...	...
tag_u	c_{u1}	c_{u2}	...	c_{uv}

ith row and jth column indicates the frequency of tag i appearing at level j throughout the m topic hierarchies. Based on the tag-level matrix, we define a weight function for each tag i at level j as follows.

$$weight(t_i, l_j) = c_{ij}^2/(\Sigma_i c_{ij} \Sigma_j c_{ij}) \tag{3.2}$$

The weight has two factors. The first one $c_{ij}/\Sigma_i c_{ij}$ captures how likely different tags are chosen for level j while the second one $c_{ij}/\Sigma_j c_{ij}$ captures how likely different levels contain tag i. The weight chooses tags for each level with the intuition that the larger the weight for tag i and level j, the more likely the tag appears at that level. In algorithm 2, we normalize $weight(t_i, l_j)$ for all tags at each level to select the most likely tags hierarchically. The algorithm computes $weight(t_i, l_j)$ using the tag-level matrix built on the m topic hierarchies from line 1 to 6. Then, starting from the first level, it chooses tags probabilistically for each level in turn from line 7 to 13. Note that by drawing tags without replacement, we disallow one tag to be chosen for multiple levels (line 9 to 12). After drawing tags for the $n_0 - 1$ levels, we leave the unused tags in T_{unused} for the last level directly (line 13).

Algorithm 2: The Topic Hierarchy Merging Algorithm

Input: m topic hierarchies T_1^i,T_2^i,...,$T_{n_i}^i$ ($i \in \{1, 2, ..., m\}$), the number of levels n_0 ($n_0 \leq \max(n_1, n_2, ...n_m)$) and the number of tags s_l ($l \in \{1, 2, ..., n_0\}$) at each level of the resultant tag hierarchy

Output: a tag hierarchy $\mathcal{H}$: T_1^0,T_2^0,...,$T_{n_0}^0$

```
begin
    let T_1^0 = T_2^0 = ... = T_{n_0}^0 = ∅;
    build the tag-level matrix M using the m topic hierarchies;
    let T_unused = T_all = {1, 2, ...} be the set of all tag indexes in M;
    let u = |T_all|, v = max(n_1, n_2, ...n_m);
    compute weight(t_i, l_j) based on M according to Eq. 3.2 for all i and j where
    1 ≤ i ≤ u, 1 ≤ j ≤ v;
    foreach level l ∈ {1, 2, ..., n_0 − 1} do
        while |T_l^0| < s_l do
            normalize weight(t_i, l) such that Σ_i weight'(t_i, l) = 1 for each
            i ∈ T_unused;
            draw a tag i according to the normalized probabilities;
            |T_l^0| = |T_l^0| ∪ {i};
            T_unused = T_unused \ {i};
    let T_{n_0}^0 = T_unused;
end
```

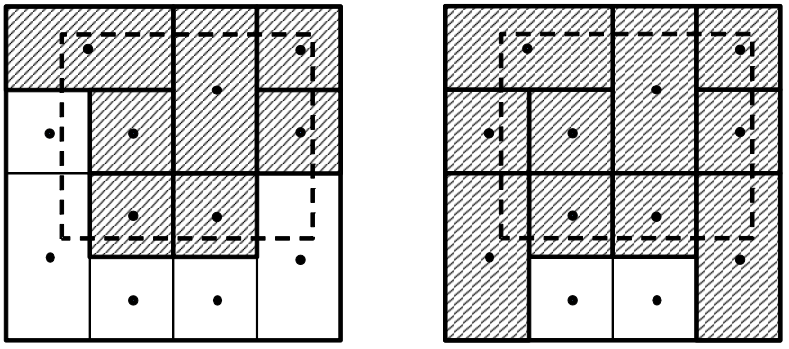
3.4.4 Ranking Merged Biclusters

Although the number of biclusters decreases after merging, there are still many biclusters due to the abundance of various topics emerging in social media. Besides, users may want to see the most interesting summaries, e.g., those discussed by more people or providing more information. Below we propose a score function so that users can rank all the biclusters to find the most interesting ones according to their preferences by simply tuning a single parameter.

$$Score(T, C) = den(T, C) \cdot \log(|T||C|) \cdot (|C|/|T|)^p \tag{3.3}$$

where $|T|$ and $|C|$ are the numbers of tags and contents respectively in a bicluster (T, C) and p is an integer tuning parameter. In the score function, the first two factors $den(T, C)$ and $\log(|T||C|)$ indicate that biclusters with larger density or size would be ranked higher. The third factor $(|C|/|T|)^p$ incorporates users' preferences by tuning $p \in \{0, \pm 1, \pm 2, ...\}$. For instance, users can set $p = 1$ in favor of biclusters with more contents or set $p = -1$ in favor of biclusters with more tags. The larger the absolute value of p, the stronger users stress their preferences. However, the absolute value of p should not be very large to avoid the dominance of the third factor.

3.4.5 Mismatch Problem



(a) Case one (b) Case two

Figure 3.4: Two cases of the mismatch problem

Recall that biclusters are merged when the corresponding partitions fall into the user-specified spatiotemporal parameters R_{tim} and R_{geo}. Since one partition can only be associated with a certain date, it is easy to check whether a partition falls within R_{tim}. Next we consider two different cases determining whether a partition falls in R_{geo} (dashed boxes in Figure 3.4): (1) the centroids of the partitions are within R_{geo} and (2) the partitions overlap R_{geo}. Both cases can lead to mismatch problem. In case one, biclusters in the shadowed partitions are merged because their centroids fall in R_{geo}. For instance, although the top-left partition is not entirely included in R_{geo}, all

the biclusters in it are merged with other partitions (false positive). Similarly, although part of the bottom-left partition is included in R_{geo}, none of its biclusters are merged (false negative). In case two, biclusters in the shadowed partitions overlapping R_{geo} are merged.

We cannot avoid the mismatch problem because a bicluster formed by various contents in a partition has no geo-coordinates in itself. Thus merging should be performed on the basis of partitions rather than biclusters. However, we can evaluate the degree of the mismatch for quality control and use it as feedback to adjust the partitioning parameter δ_{cnt}.

Definition 3.6. *Given a user-specified geographical range R_{geo}, the **mismatch rate** $rate_{mis}$ of the PM scheme is defined as the ratio of num_{mis} to num_{tp}, where num_{mis} is the number of mismatched biclusters and num_{tp} is the number of biclusters covered by R_{geo}.*

Based on the above definition, the mismatch rate can be larger than 1, which is indicative of a severe mismatch. Next we discuss how to compute the mismatch rate for the two cases in Figure 3.4.

Firstly, we consider case one in Figure 3.4(a). Partitions covered by R_{geo}, either partially or entirely, form three different sets P_{fp}, P_{fn} and P_{en}. The first two contain partitions partially covered by R_{geo}: P_{fp} contains those whose centroids fall in R_{geo} while P_{fn} contains those whose centroids do not. P_{en} has partitions entirely falling in R_{geo}. For partitions in P_{fp} and P_{fn}, we estimate the number of mismatched biclusters according to the coverage rate of the partitions as follows.

$$\begin{aligned} num_{fp} &= \textstyle\sum_{pa \in P_{fp}} num(pa) \cdot (1 - cov(pa)) \\ num_{fn} &= \textstyle\sum_{pa \in P_{fn}} num(pa) \cdot cov(pa) \end{aligned} \tag{3.4}$$

where $num(pa)$ is the number of biclusters in partition pa and $cov(pa)$ is the percentage of pa covered by R_{geo}. Thus num_{fp} is the estimated number of additional biclusters (false positives) used in the merging phase while num_{fn} is the estimated number of missed biclusters (false negatives) falling in R_{geo} but omitted in the merging phase. Next we compute the estimated number of biclusters in R_{geo} by

$$num_{tp} = \sum_{pa \in P} num(pa) \cdot cov(pa) \tag{3.5}$$

where $P = P_{fp} \cup P_{fn} \cup P_{en}$. With the above equations, we can evaluate the quality of the PM scheme by approximating the mismatch rate as follows.

$$rate_{mis} = \frac{num_{mis}}{num_{tp}} = \frac{num_{fp} + num_{fn}}{num_{tp}} \tag{3.6}$$

For case two in Figure 3.4(b), P_{fn} is empty because biclusters in any partition overlapping the geographic region R_{geo} are merged. However, the false positive partition set, denoted by P'_{fp}, now contains all partitions partially overlapping R_{geo}, including those with centroids falling outside R_{geo}. The mismatch rate in this case is as follows.

$$rate_{mis} = \frac{num'_{fp}}{num_{tp}} = \frac{\sum_{pa \in P'_{fp}} num(pa) \cdot (1 - cov(pa))}{num_{tp}} \tag{3.7}$$

3.5 System Implementation

Based on the PM scheme, we build a system called Vesta to browse and explore the top-ranked summaries interactively.

3.5.1 System Architecture

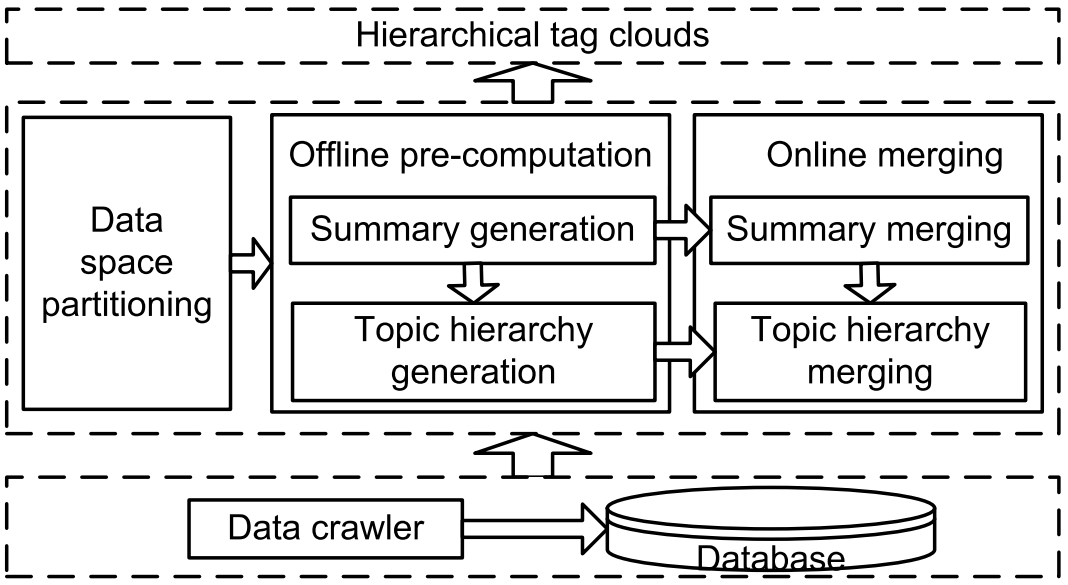


Figure 3.5: System architecture

Figure 3.5 shows our system architecture, consisting of data crawling, processing and visualization components from bottom to top. The data crawling component keeps crawling geo-coded social media contents, cleaning and storing the data in the database. The processing component is the core of the system, including three parts which perform data partitioning, offline pre-computation and online merging respectively. The data partitioning part starts splitting the data into partitions after the crawler finishes preparing the contents of each day. Offline pre-computation is then performed to generate summaries and topic hierarchies for the partitions. Note that our PM scheme makes it possible that the pre-computation can be done for different partitions in parallel. The summaries and topic hierarchies in corresponding partitions will be merged respectively in the online merging part once users specify the spatiotemporal parameters. The merged summaries are finally passed to the visualization component to be displayed as hierarchical tag clouds according to their corresponding tag hierarchies generated by merging topic hierarchies.

Although our system performs the partitioning and pre-computation on a daily basis which leads to a one-day delay, we can reduce the delay to make Vesta a semi-realtime system. For instance, we can process the data every 3 hours and merge the results gradually until all contents of the day is processed.

3.5.2 Visual Layout

Next we illustrate the layout of the hierarchical tag clouds in the visualization component, which is built based on Google Maps[2] and D3[3]. Recall the example in Figure 3.1, the first-level tags of the hierarchical tag clouds will be displayed first in the largest font size. As users zoom in, tags at subsequent levels of each tag cloud will be placed near and around their first-level tags in smaller size level by level. To reduce the possibility of overlap among tags in different tag clouds and leave enough space to arrange tags of the same tag cloud close to each other, we scatter the first-level tags evenly in two concentric circular orbits. The radii of the two orbits can be adjusted to reduce the overlap among tag clouds as much as possible. Collision detection is performed to help determine the positions of the tags at subsequent levels when they are placed around the corresponding first-level tags.

3.6 Experimental Study

In this section, we conduct an experimental study to assess the proposed methods from different perspectives.

[2]https://maps.google.com/
[3]A JS library for data visualization (http://d3js.org/).

3.6.1 Data Sets and System Environment

The experiments are conducted over the real-world geo-coded tweets crawled using the Twitter Streaming API[4]. Each tweet is associated with a pair of latitude-longitude coordinates and a time. On average, we can receive 4.1M to 4.3M (million) geo-coded tweets every day, among which about 0.7M are in English. The four data sets used in the experiments contains 0.1M, 0.7M, 2.5M and 4.3M tweets respectively. As around 1% of tweets are geo-coded[5], 4.3M is a proper approximation of the number of geo-coded tweets published daily, given that 400M to 500M tweets are generated altogether per day.

All the source code is written in Java, and the Twitter data is stored in MySQL 5.1.60. The experiments were conducted on Windows Server 2003 Enterprise x64 Edition with 16-core 2.29GHz CPU, 64GB RAM and JRE6.

3.6.2 Comparison of Different Summarization Methods

Firstly we compare our methods *ours_tag* and *ours_content* based on FCA with two other biclustering methods (*OABicluster* [30] and *FLOC* [94, 93]) and two topic modeling methods (*LDA* [15] and *hLDA* [13, 12]). *OABicluster* was also proposed for analyzing social media data based on FCA. It generates formal concepts for each tag-content pair, without guaranteeing the bicluster quality. *FLOC* was originally proposed for expression data analysis. It first generates k initial biclusters randomly and then improves their quality iteratively, by allowing missing values which correspond to the sparsity in social media content data. It uses the residue and volume (i.e., the number of non-empty values [93]) in a gain function to measure the bicluster quality. To make *FLOC* comparable, we replace the residue in the gain function with density instead, which does

[4]https://dev.twitter.com/docs/streaming-apis
[5]http://www.scotthale.net/blog/?p=307

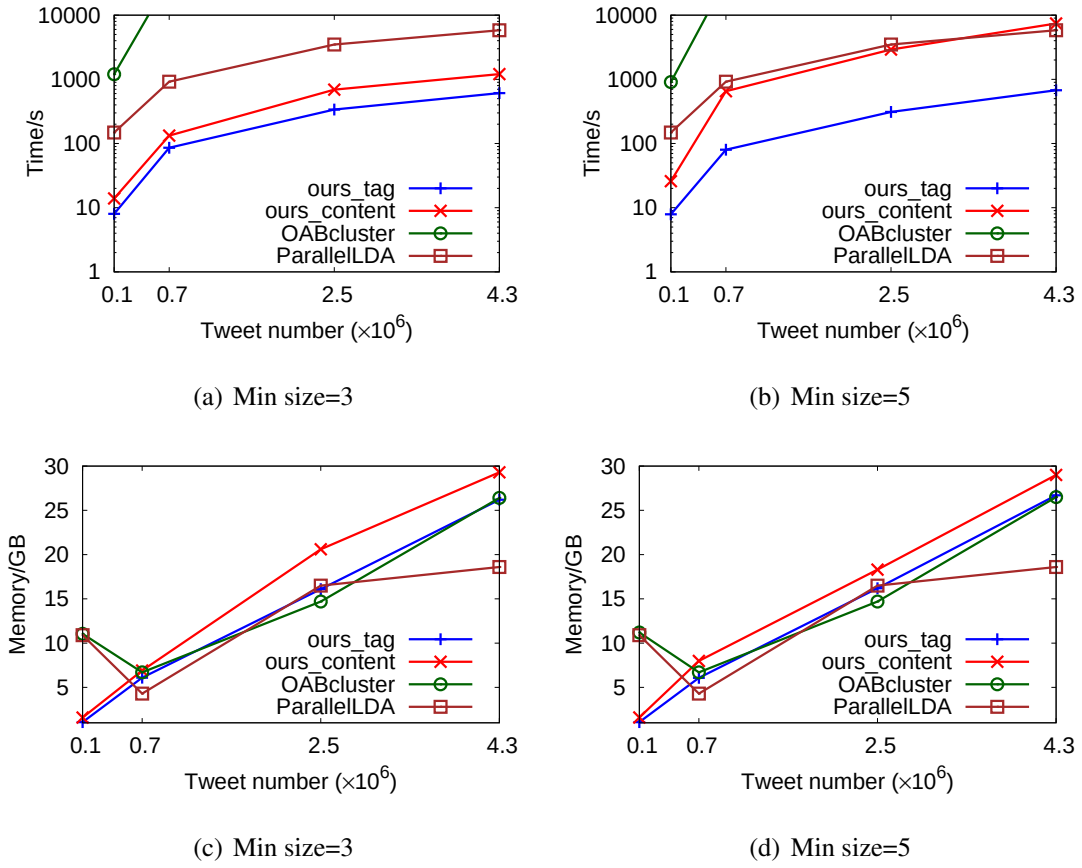
not affect the main procedure. The two topic modeling methods generate topics which can also be perceived as summaries. We use MALLET[6] which has implemented *ParallelLDA* and *hLDA*, where *ParallelLDA* is a parallel version of *LDA* to accelerate the speed.

For the first four biclustering methods, we set the minimum density δ_{den} from 0.5 to 1.0 and the minimum size δ_{sz} from 3 to 5. Since different values of δ_{den} do not affect the performance obviously, we only report the results with δ_{den} equal to 0.8 and δ_{sz} equal to 3 and 5. Note that δ_{den} and δ_{sz} do not apply to *ParallelLDA* and *hLDA*. The execution time and memory usage of *ParallelLDA* in Figure 3.6 are duplicated for different δ_{sz} for comparison purpose only. For *ParallelLDA*, we set the topic number to 100 and thread number to 4. For *hLDA*, we set the level number of the topic hierarchy to 5. The iteration numbers for both are set to 1000.

Performance

Figure 3.6 shows the performance of the above methods over four data sets with different number of tweets. We terminated *FLOC* and *hLDA* because they cannot finish within 10 hours for any data set. They run slowly due to the iterative nature. The density and volume parts of the gain function also decelerate *FLOC* probably because the density part tends to shrink the bicluster size while the volume favors larger size. To accelerate *FLOC* and make the densities of the initial biclusters increase faster, we remove the volume part and denote the modified *FLOC* by *FLOC_modified*. Although we see some improvements, *FLOC_modified* still cannot finish within 10 hours. *hLDA* is even slower and cannot finish the first 10 of 1000 iterations within 10 hours. Although we show later that *hLDA* works for the offline topic hierarchy generation in our system where it usually handles only thousands of tweets for a partition, *hLDA* in itself cannot

[6]A java package for machine learning (http://mallet.cs.umass.edu/).



(a) Min size=3

(b) Min size=5

(c) Min size=3

(d) Min size=5

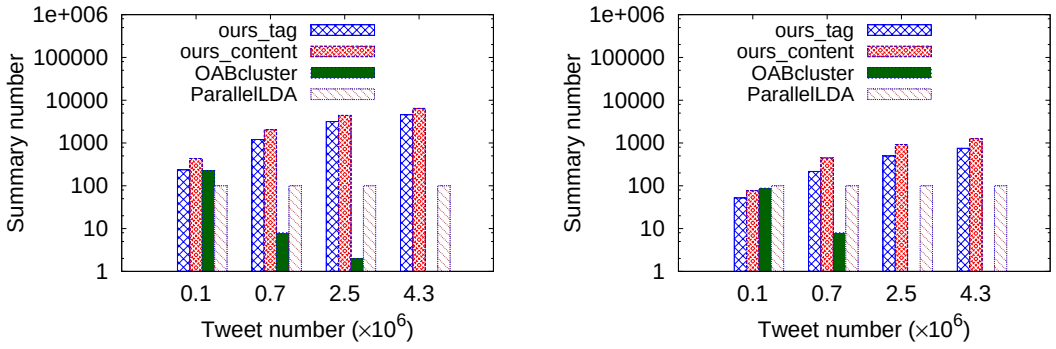
Figure 3.6: Performance comparison

scale to larger data sets containing hundreds of thousands of tweets. Thus *hLDA* cannot be applied to large amounts of social media data for topic hierarchy generation directly. We do not report the results for *FLOC*, *FLOC_modified* and *hLDA* here.

Figure 3.6(a) and 3.6(b) show the execution time of the other four methods. When δ_{sz} is 3, our proposed methods *ours_tag* and *ours_content* outperform *ParallelLDA* by almost an order of magnitude. When δ_{sz} is 5, the performance of *ours_tag* almost remains unchanged while the execution time of *ours_content* approaches that of *ParallelLDA* for tweet number larger than 0.1M. Note that *ParallelLDA* is set to generate only 100 topics. It will take more time if generating as many as *ours_content* does. For both values of δ_{sz}, *OABicluster* can only finish running within 10 hours when the tweet number is

0.1M.

Figure 3.6(c) and 3.6(d) show the memory usage. Almost all the methods consume more memory when the tweet number gets larger, although some drops are observed for *OABicluster* and *ParallelLDA* when the tweet number comes to 0.7M. *ParallelLDA* uses less memory than other methods when the tweet number is 4.3M. This is probably because we set the topic number to 100 which is a relatively small value. The memory usage of our proposed methods increases linearly, which brings up concerns that they may not fit into memory given even larger data sets. We will show later that our PM scheme converts this memory-based problem to a disk-based problem so that memory usage becomes manageable.



(a) Min size=3 (b) Min size=5

Figure 3.7: Summary detection capability comparison

Summary Detection Capability

Figure 3.7 shows the number of biclusters/topics each method can generate, which to some extent reflects their capability of detecting various summaries. Since *OABicluster* failed to finish within 10 hours for the other three data sets, we report the number of biclusters it had generated when we terminated it. We omit *FLOC* and *FLOC_modified* again since they cannot finish running within 10 hours for any data set. Upon termina-

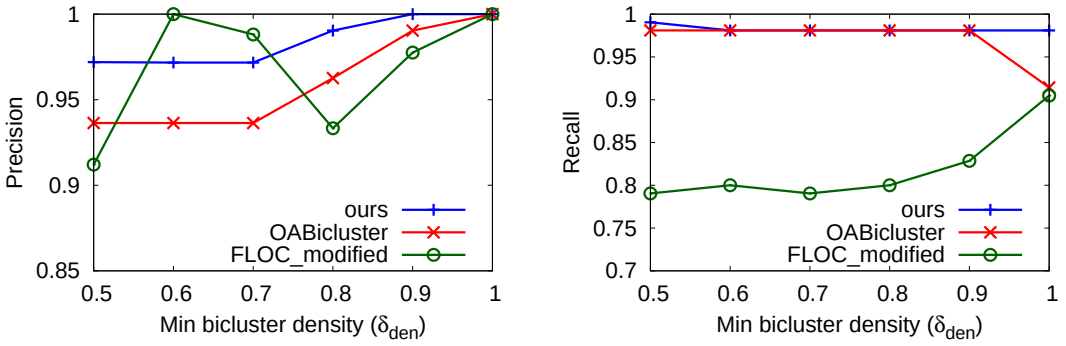
tion, *FLOC* generated no bicluster with density larger than 0.5 while *FLOC_modified* only generated several biclusters with density larger than 0.8 for the data set having 0.1M tweets. As the number of tweets increases, it is intuitive that more summaries or topics will be covered. Our methods conform to this intuition and generate more biclusters for larger data sets. *ParallelLDA* generates 100 topics for all data sets since we set the topic number to 100.

We also conduct experiments given δ_{den} set to 0.5 to 1.0 over the 0.1M tweets where *OABicluster* can finish running within 10 hours. *OABicluster* generates slightly more than or comparable to the number of biclusters our methods generate when δ_{den} ranges from 0.5 to 0.7. The number of biclusters generated by *OABicluster* decreases rapidly when δ_{den} increases from 0.8 to 1.0, indicating that it tends to miss many high-density biclusters. The number of biclusters generated by our methods does not drop obviously as δ_{den} increases. Because of space limitations, we omit to present the figures. In addition, both *OABicluster* and *FLOC* may lead to cases that a single bicluster covers multiple unrelated summaries for small δ_{den}. For instance, given two biclusters $(\{t_1, t_2, t_3\}, \{c_1, c_2, c_3\})$ and $(\{t_4, t_5, t_6\}, \{c_4, c_5, c_6\})$, both with density equal to 1, *OABicluster* and *FLOC* may generate a bicluster $(\{t_1, t_2, t_3, t_4, t_5, t_6\}, \{c_1, c_2, c_3, c_4, c_5, c_6\})$ with density equal to 0.5 but covering two unrelated summaries. Our methods can avoid this problem since they first generate "full-density" biclusters and related biclusters will be merged if the density of the resultant bicluster is larger than δ_{den}.

Precision and Recall

Evaluating the quality of summaries of social media contents quantitatively is often hard because of the huge number of contents and lack of predefined summaries or topics to compare against. Thus, we try to sample a small set of tweets by hand and predefine summaries using the tweets in order for the evaluation. Specifically, we manually select

82 out of the 0.1M tweets discussing 4 different topics to verify whether these methods can generate proper summaries for the topics. The tweets fall into 4 groups according to the topics they belong to. Common tags in all tweets of a group are chosen as the summary (or ground truth) of that group. 18 other randomly selected tweets are added as noise. All the 100 tweets are used to generate biclusters using different methods and the results are compared with the group truth. The number of true positive tags and tweets is divided by that of all positive tags and tweets to obtain the precision, and the former again is divided by the number of all tags and tweets in the ground truth to obtain the recall. Since *ParallelLDA* and *hLDA* cannot find the related tweets of the generated topics, we exclude them from the comparison.



(a) Precision

(b) Recall

Figure 3.8: Precision and recall

Figure 3.8 shows the results when δ_{sz} is 3 and δ_{den} varies from 0.5 to 1. Since the results of *ours_tag* and *ours_content* overlap, we denote them as *ours* for simplicity. The precision and recall of our methods are stable and larger than those of other methods for most δ_{den} values. The recall of *OABicluster* is close to ours but drops suddenly when δ_{den} comes to 1. The average precision of *FLOC_modified* is competitive, however, the precision fluctuates greatly and the recall is relatively low. Besides, *FLOC_modified* does not guarantee to find all four topics every time. We do not report *FLOC* as it often

discovers only one topic and mixes other topics together.

3.6.3 Partition-and-Merge Scheme Evaluation

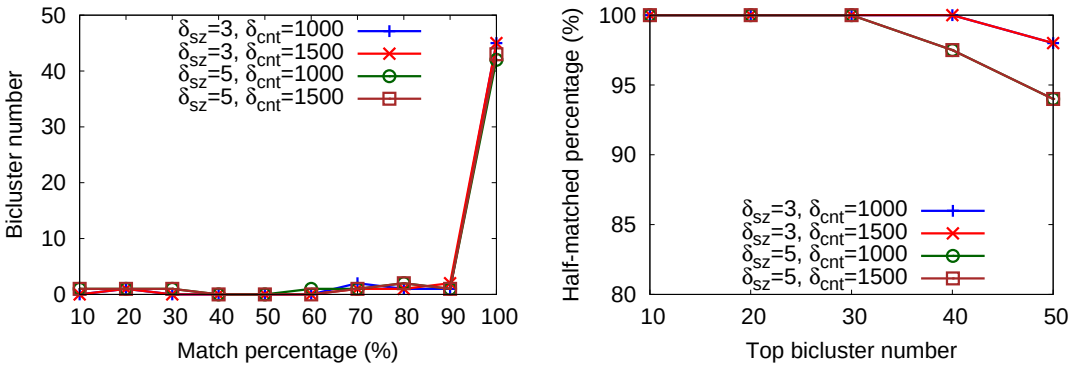
Next we evaluate the effectiveness of our PM scheme, including assumption validation and mismatch evaluation.

Assumption Validation

We validate the assumption proposed in Section 3.4.2 that globally interesting summaries are also interesting in certain partitions. In terms of biclusters, those generated without partitioning the data space and highly ranked can also be generated in some partitions of the data space. The validation can prove the validity of our PM scheme.

The validation is performed over the data set of 0.7M tweets. We first partition the data space and generate biclusters by running *ours_tag* and *ours_content* for each partition. δ_{cnt}, the largest number of contents in a partition, is set to 1000 or 1500. The biclusters in all partitions form set B_{par}. Then we generate biclusters, which form set B_{no}, directly without partitioning and test whether the most interesting biclusters in B_{no} also exist in B_{par}. To find the most interesting biclusters in B_{no}, we rank them according to our proposed score function with p set to 0. δ_{sz} for biclusters in B_{par} and B_{no} is set to the same value which is 3 or 5. We do not set δ_{den} as all biclusters in B_{par} and B_{no} are "full-density".

Figure 3.9 validates our assumption. In Figure 3.9(a), we choose the top 50 biclusters from B_{no} and compare them against biclusters in B_{par}. For each bicluster b from the top 50, we find a bicluster b' in B_{par} which shares the largest number of common tags with b. The match percentage is the ratio between the number of common tags and the number of tags in b, reflecting to what extent an interesting summary generated without partitioning can also be discovered when partitioning is performed. Figure



(a) Number of top biclusters (b) Half-matched percentage

Figure 3.9: Assumption validation

3.9(a) shows, for different parameter settings, most of the top 50 biclusters from B_{no} can find a bicluster in B_{par} which shares all the tags in the top biclusters, indicating that many of them can be rediscovered using the partitioning scheme. A high match percentage between 50% and 100% can also indicates that a summary is quite likely to be rediscovered. Figure 3.9(b) shows the percentage of the top k biclusters that have match percentage more than 50% when k is set to 10, 20, 30, 40 and 50. All the top 30 biclusters have match percentage more than 50%. As k increases, the half-matched percentage begins to decrease. However, there are still more than 90% of the top 50 biclusters whose match percentage is over 50%, meaning summaries associated with more than 45 out of 50 biclusters are highly likely to be discovered using the partitioning scheme. We also note that the results would be even better if biclusters in B_{par} are merged.

Mismatch Evaluation

Below we take a fixed geographic region containing 0.1M tweets as an example to explain the mismatch problem. Figure 3.10 shows the mismatch rate $rate_{mis}$ and bicluster number as δ_{cnt} varies when $\delta_{sz} = 3$ and $\delta_{den} = 1$. Note that $rate_{mis}$ for

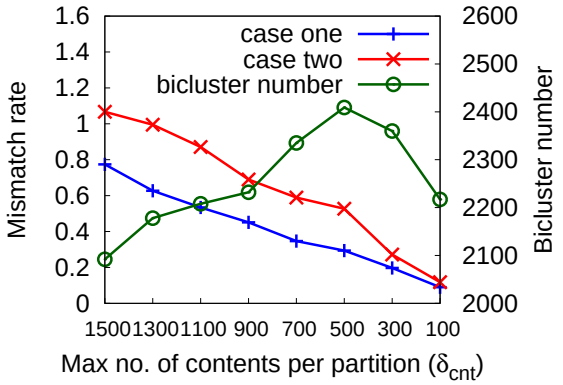


Figure 3.10: Mismatch evaluation

either case of the mismatch in Figure 3.4 can be larger than 1, indicating that a severe mismatch occurs.

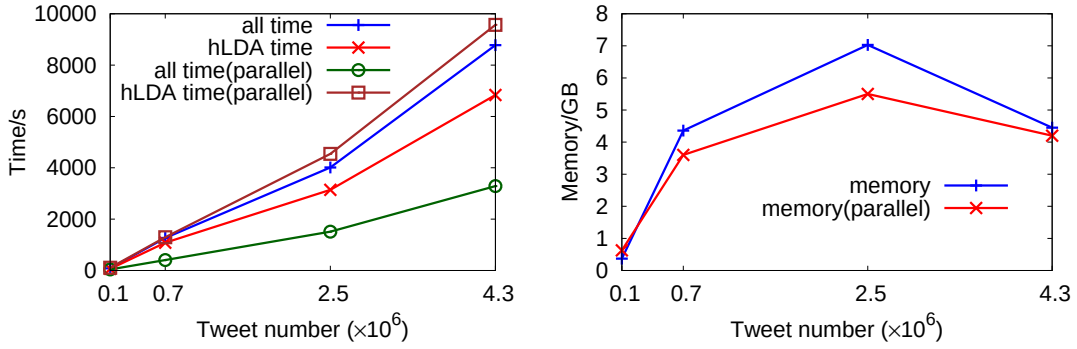
From Figure 3.10, we can see that $rate_{mis}$ for either mismatch case decreases as δ_{cnt} drops and that the first case leads to a smaller $rate_{mis}$. There are three stages for the decrease, namely when δ_{cnt} is larger than 700, between 700 and 500, and smaller than 500, where more obvious is the second stage. When δ_{cnt} decreases from 1500 to 700, $rate_{mis}$ reduces in a relatively fast speed, which means that any value between 1500 and 700 may not be proper for δ_{cnt} since $rate_{mis}$ is not stable enough. When δ_{cnt} comes from 700 to 500, $rate_{mis}$ decreases less dramatically, indicating it is a reasonable value range for δ_{cnt}. After δ_{cnt} drops below 500, $rate_{mis}$ decreases rapidly again. The instability of $rate_{mis}$ in this stage is probably because δ_{cnt} becomes too small. A very small $rate_{mis}$ is not always meaningful. It may cause loss of biclusters due to very small partitions, which is reflected by the green bicluster number line in Figure 3.10. Another strategy is to choose a value, 500 in Figure 3.10 for instance, for δ_{cnt} when the number of biclusters reaches its peak. This is also consistent with choosing between 700 and 500, and more practical in case the second stage is less detectable. Consequently, users do not have to set δ_{cnt} blindly by themselves. As δ_{cnt} can be determined automatically by conducting the mismatch evaluation, different values of δ_{cnt} can be set for different

geographic regions adaptively.

3.6.4 System Scalability Analysis

Vesta adopts the PM scheme which converts the in-memory biclustering to a disk-based approach and makes it possible to parallelize the approach. Next we analyze the system scalability for the partitioning and merging phases.

Offline Scalability



(a) Time

(b) Memory

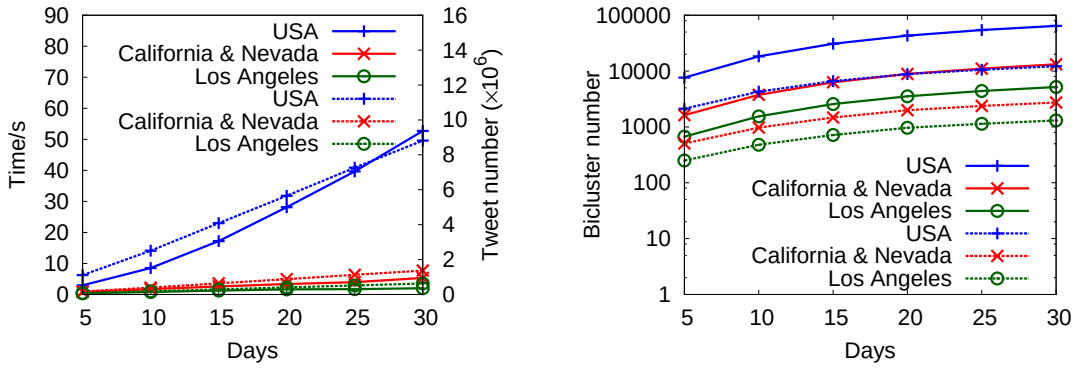
Figure 3.11: Offline scalability

The offline partitioning phase includes data space partitioning and pre-computation of biclusters and topic hierarchies. We plot the execution time and memory usage over different data sets in Figure 3.11 when δ_{sz}, δ_{cnt} and the number of iterations for *hLDA* are set to 3, 1000, 1000 respectively. The blue line shows the execution time increases almost in proportion to the data set size while most of the time is consumed by *hLDA* (the red line) to generate topic hierarchies. The green line indicates a significant improvement when we parallelize the partitioning phase using 4 threads. Because we simply add up the time consumed by *hLDA* in all threads without considering the overlap, the *hLDA* time under parallelism (the brown line) gets very large. This indicates

that *hLDA* can be parallelized to greatly accelerate the process. Through parallelism, the overall time of the partitioning phase reduces from two and a half hours to less than an hour for the data set with 4.3M tweets, which approximate the number of geo-coded tweets published every day in reality.

Figure 3.11(b) shows the memory usage which decreases greatly compared with that in Figure 3.6(c) and 3.6(d). The memory usage does not always increase as the data set gets larger. Note that the amount of memory recorded here is the peak value which is mainly caused by *hLDA* when faced too many tweets as input. That value often drops after a short while during execution. Again, the parallelism also leads to less memory usage. Thus Figure 3.11 indicates that the offline partitioning phase can scale to even larger data sets especially through parallelism.

Online Scalability



(a) Time

(b) Bicluster number decrease

Figure 3.12: Online scalability

The online merging phase is done at runtime when users specify the spatiotemporal ranges. The solid lines in Figure 3.12(a) show the response time for three geographic areas on different scales w.r.t. various time ranges. The dotted lines show the corresponding number of tweets involved in certain ranges. δ_{den} is set to 0.5, which is the

lower bound density of merged biclusters. Given a geographic area, the response time increases linearly with the number of days (also the number of biclusters to merge or tweets). The response time is often small (e.g., 2 seconds for L.A. or 5 seconds for California & Nevada over 30 days) when the geographic area is on a city or state scale. When it comes to countries such as the entire USA, the response time is relatively longer because of the huge number of tweets.

Although the number of biclusters to merge is often large, that of the merged biclusters decreases greatly. This is exhibited in Figure 3.12(b) where solid (or dotted) lines represent the number before (or after) merging, showing that this number can be reduced by $63\% \sim 81\%$. It means that a large part of biclusters from different partitions can be merged together and thus validates again the effectiveness of our PM scheme.

3.7 Summary

In this chapter, we proposed a system called Vesta which enables interactive exploration of different regions by summarizing and browsing social media contents via hierarchical tag clouds. We proposed to generate summaries by biclustering the contents based on FCA and then extended the approach by introducing a disk-based PM scheme for better scalability. For visualization purposes, we adopted hLDA to generate topic hierarchies and merge them to form a tag hierarchy for each summary. The experimental study demonstrated the efficiency and effectiveness of our methods.

CHAPTER 4

HIERARCHICAL SUMMARIZATION OF SOCIAL MEDIA CONTENTS BASED ON DBPEDIA ONTOLOGY

4.1 Overview

In Chapter 3, we proposed Vesta to generate summaries without taking into account semantic meanings and relationships among keywords in a summary. To explore the meanings of massive amounts of social media data, in this chapter we propose Heron[1], a novel summarization approach to generating summaries with clear and coherent semantic meanings from social media contents. However, unlike most existing summarization methods, our approach generates summaries by grouping entities with close relationships according to the classes to which the entities are mapped in the *DBpe-*

[1]Heron stands for *hier*archical summarizati*on*.

dia ontology[2] [4, 11, 57]. A hierarchical structure is also built to reveal fine-grained subsumptive relationships among subsets of entities in each summary. In this case, a summary generated by our approach is also called a *hierarchical summary*, because it not only consists of closely related entities but also has a hierarchical structure, which is a sub-hierarchy of the DBpedia ontology, to capture subsumptive correlations among the entities.

Since the entities in a summary are Wikipedia entities extracted from social media contents, and the DBpedia ontology plays an important role in our approach, we next provide a short introduction to them before elaborating on the DBpedia ontology based summarization.

4.1.1 Wikipedia Entity and Infobox

Wikipedia[3], as one of the most popular multilingual Internet encyclopedias, has already had 18 billion page views and nearly 500 million unique visitors every month as of February 2014[4]. The English Wikipedia has over 4 million articles, remaining the largest among all the Wikipedias in over 200 languages[5]. Generally, each Wikipedia article (except the functional pages such as talk pages, redirects, etc.) corresponds to a *Wikipedia entity*, or *entity* for short, which is often represented using the title of the article. For instance, the Wikipedia article (https://en.wikipedia.org/wiki/White_House) entitled "White House" corresponds to the entity White_House, where the words within an entity are connected by underscore(s) for easier processing.

In some of the Wikipedia articles, especially those that are popularly viewed and actively edited, an *infobox* is added to each article to represent a summary of information

[2]http://wiki.dbpedia.org/Ontology39
[3]https://www.wikipedia.org/
[4]http://www.nytimes.com/2014/02/10/technology/wikipedia-vs-the-small-screen.html
[5]https://en.wikipedia.org/wiki/English_Wikipedia

regarding the corresponding entity [98]. An infobox is a piece of structured information which is located in the top-right corner of an article, having a class label (indicating to which class an entity belongs) and a group of property-value pairs. An infobox can be created easily by choosing one of the various infobox templates and filling values of properties in the selected template. In the English Wikipedia alone, more than 6,000 infobox templates have been created and reused by Wikipedia contributors [68]. This crowdsourcing nature of Wikipedia inevitably brings about great challenges in terms of the consistency and accuracy of how the templates are created and used. After all, determining a proper template from among thousands of options is challenging enough for most contributors. For instance, different contributors tend to choose different yet related templates for the same entity, and may use different property names to describe the same property in different templates [11].

4.1.2 DBpedia Ontology

Because of the inconsistency and redundancy in Wikipedia, the DBpedia project[6] was initiated in 2007, with the aim of extracting structured content from vast amounts of information (e.g., infoboxes) in Wikipedia. DBpedia allows users to issue sophisticated queries against Wikipedia, and to link various data sets to Wikipedia. As of July 2014, The English version of the DBpedia knowledge base (DBpedia 3.9) describes 4.0 million Wikipedia entities, of which 3.22 million are classified in a consistent ontology[7], i.e., the DBpedia ontology. This ontology has been created manually based on the most frequently used infoboxes in Wikipedia, and covers 529 classes, which are described by 2,333 different properties. Classes in the ontology form a subsumptive hierarchy.

[6] http://dbpedia.org/About
[7] https://en.wikipedia.org/wiki/DBpedia

Figure 4.1: DBpedia ontology (partial)

Figure 4.1 shows part of the DBpedia ontology[8]. The classes in the ontology form a tree-like hierarchy, which is rooted at a general node "owl:Thing". Note that "owl:Thing" itself is not a class in the ontology, and thus we say that it is at level 0 and that the children of it (e.g., "Activity" and "Agent") are at level 1. All the classes are organized as either internal or leaf nodes in the hierarchy, where classes at higher levels represent more general meanings and are superclasses or ancestors of those appearing in the sub-hierarchies rooted at these classes. For instance, "Activity" is the superclass of "Sport", which is again the superclass of "Athletics", "BoxingStyle", etc.

[8]The snapshot is taken from the ontology classes page at http://mappings.dbpedia.org/server/ontology/classes.

4.1.3 DBpedia Ontology Based Summarization

By making use of the class information of Wikipedia entities, we are able to map the entities appearing in social media contents to the DBpedia ontology. Because classes in the ontology form a hierarchical structure which captures the subsumptive relationships among the classes, entities mapped to the ontology will thus be naturally grouped together according to where their corresponding classes are located in the ontology. Specifically, entities having the same class label will be mapped to the same class node in the ontology hierarchy, while those whose classes have certain parent-child (or ancestor-descendant) relationships will also be mapped to reserve those relationships. Therefore, we can "cluster" the mapped entities conveniently so that each "cluster" of entities serves as a summary of their corresponding social media contents.

Such a summary not only consists of a group of related entities, but also captures and reserves a hierarchical structure of them in terms of subsumptive semantics. In this manner, we are able to enrich a summary in two respects. On the one hand, although entities in a summary are closely related, we categorize them into several subsets so as to better depict the similarities as well as the subtle distinctions among entities in the same summary. The hierarchical relationships among the subsets, on the other hand, represent the summary from a one-dimensional space (with only entities) to a two-dimensional space (with both entities and correlations), to reflect the interconnections among subsets of entities within a summary.

Example 4.1. *Figure 4.2 shows an example of a hierarchical summary. The summary consists of entities (e.g.,* Coral *and* Guava*) belonging to the eukaryote class, one of the major biological domains of organisms. The entities in the summary are divided into more specific subsets, and the subsets further form a hierarchical structure according to the relationships among the subsets of entities. Specifically, on the left is the hierarchy of subsets of entities, while on the right is the corresponding sub-hierarchy of the DBpedia*

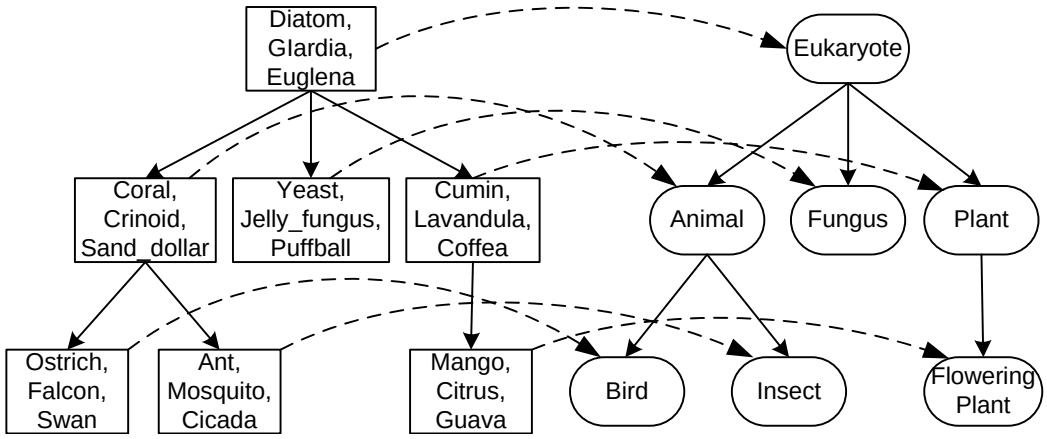


Figure 4.2: An example of a hierarchical summary

ontology, based on which the left part is constructed. Each subset of entities is mapped to a class node in the sub-hierarchy, as indicated by the dashed arrows. For instance, a dashed arrow goes from the subset having entities Ant, Mosquito and Cicada to the class "Insect", meaning that the three entities are classified as insect.

In addition to the revelation of semantic correlations within each summary, summaries generated from different data sets can also be utilized for the comparison of the data sets (e.g., tweets about China and tweets about the USA) in terms of entity distribution. For instance, users can generate a summary corresponding to the same sub-hierarchy in Figure 4.2 using a set of tweets about China and another set about the USA respectively. They can compare the two summaries to discover which eukaryotic entities are frequently mentioned along with "China" and "USA", and how they are distributed in each set of tweets. This can be achieved because the entities are mapped to a central and stable ontology.

Another advantage of the proposed hierarchical summarization is that the summary is quite convenient for visualization. We have entered the era of Big Data, which means that the volume of data for processing is quite huge. It is therefore vital to present the data properly and vividly, and to make it easily understandable. Tag cloud [10, 24,

78] is an appropriate and popular form to visualize entities, which are called tags in a tag cloud. Tags are often rendered in different colors, font sizes and types to convey certain meanings, such as frequency, importance, recency and so on. Recall that in the previous chapter we proposed hierarchical tag clouds [40] for the visualization of summaries extracted from spatiotemporal social media contents. The differences are that those summaries are not composed of Wikipedia entities and that the hierarchies of tags are generated based on some statistical method, thereby lacking a clear meaning of the relationships among the tags. With enlightenment from these works, however, the hierarchical summaries proposed in this chapter can be visualized in hierarchical tag clouds to convey clearer semantic relationships among the entities in each summary.

In this chapter, we propose Heron to generate hierarchical summaries for social media contents, which are capable of grouping entities in closely related classes and revealing their interconnections by leveraging the DBpedia ontology. To accomplish our goal, we first need to extract entities from the given social media contents and map those entities onto the DBpedia ontology. As mentioned earlier, however, inconsistency and inaccuracy widely exist in Wikipedia infoboxes, which renders the mapping of entities and the subsequent summarization of social media contents inaccurate. Furthermore, the presence of thousands of properties in total and the low degree of property overlap among entities also bring great challenges to traditional classification methods of reclassifying the entities. To address this problem, we propose a model named *multi-level Naive Bayes Classifiers* (mNBC) to refine the entities' classes effectively before mapping them. Besides the refinement, our model can also be used to make predictions/classifications for new entities to aid the addition of infoboxes to Wikipedia, as long as the properties are provided.

After entities are mapped to the DBpedia ontology, we then split it to generate hierarchical summaries. The process is tunable to enable the granularity of the summaries

to be controlled. Since we take advantage of the structure of the DBpedia ontology for the summary generation, the hierarchical relationships among entities are naturally reserved within each summary. Note that in the mapping step, a great number of entities might be mapped to a single class node. We further propose a ranking procedure for the selection of the most important and relevant entities according to a score formula.

To sum up, we make the following contributions in this chapter: (1) We propose to generate hierarchical summaries based on the DBpedia ontology to introduce semantics into each summary. (2) To reduce the propagation of inconsistency and inaccuracy in Wikipedia, we present a model named multi-level Naive Bayes Classifiers to refine the classes of entities before mapping them. (3) We also propose a ranking procedure to select the most relevant entities for each class.

The rest of this chapter is organized as follows. We first discuss the preliminaries and problem definition in Section 4.2. In Section 4.3, we propose the model of multi-level Naive Bayes Classifiers and present the refinement of entities' classes based on the model. In Section 4.4, we introduce the generation of summaries and the selection of top entities. We then present the experimental study in Section 4.5. Lastly, we provide a summary of this chapter in Section 4.6.

4.2 Preliminaries

In this section, we introduce some relevant concepts, followed by a formal problem definition.

Definition 4.1. *Given an entity e which is classified as $c_j \in \mathcal{C}$ ($\mathcal{C}$ is the collection of all classes in the DBpedia ontology). A sequence of classes $c_i, c_{i+1}, ..., c_{j-1}, c_j$ $(i \leq j)$ is called the* ***class chain*** *of e if (1) c_i is a class at the first level of the DBpedia ontology, (2) c_p is the parent of c_{p+1} $(i \leq p < j)$ when $i \neq j$.*

The class chain of an entity actually consists of all the classes along the path in the DBpedia ontology from the root node to the class of the entity. For instance, since the class of the entity Backstreet_Boys is "Band", the class chain of it will be "Agent, Organisation, Band". Note that a class chain not only contains a set of classes but also reserves the order of the classes, from the topmost level to the lowest as in the ontology. Although an entity can be classified as any class in its class chain, we often refer to the last one (e.g., "Band" in the above example) whey saying that an entity is classified as a class because the last one is the most specific and accurate class for the entity.

Definition 4.2. *Given a set E of entities and a set $C \subset \mathcal{C}$ of classes where all the classes in C corresponds to a sub-hierarchy h of the DBpedia ontology. If E can be divided into $|C|$ (i.e., the cardinality of C) subsets $E_1, E_2, ..., E_{|C|}$ (some subsets may be empty), where $E_i \cap E_j = \emptyset$ $(i \neq j)$ and $\cup_{i=1}^{|C|} E_i = E$ for any $i, j \in \{1, 2, ...|C|\}$, such that some injective function $f(\cdot)$ can map each nonempty subset to a unique class in C (i.e., $E_i \mapsto c_i$ and $E_i \neq \emptyset$ for $i \in \{1, 2, ..., |C|\}$), we say that the combination of all the subsets of entities, the set of classes and the injective function defines a **hierarchical summary**, or **summary** for short. A summary is denoted by $\mathcal{S}(\{E_1, E_2, ..., E_{|C|}\}, C, f(\cdot))$[9], where each nonempty subset E_i $(i \in \{1, 2, ...|C|\})$ is mapped to a class $c_i \in C$ in the sub-hierarchy h according to $f(\cdot)$.*

Note that $f(\cdot)$ in this definition is an injective function that maps any nonempty subset $E_i \subset E$ of entities to a class $c_i \in C$ because all the entities in E_i are classified as c_i. The function is injective in that it never maps distinct subsets to the same class in C. In other words, each class has at most one subset mapped to it. Intuitively, this is because entities mapped to a class in the sub-hierarchy are all grouped into one single subset.

According to the above definition, a summary $\mathcal{S}(\{E_1, E_2, ..., E_{|C|}\}, C, f(\cdot))$ defined

[9]For simplicity, we also denote a summary by $\mathcal{S}$ in this chapter.

in this chapter therefore not only consists of several subsets of entities, but also has a hierarchical structure which connects all the subsets. The hierarchical structure of a summary is actually a sub-hierarchy of the DBpedia ontology, which is determined by the classes in C. Next, we summarize our goals and define the problem to tackle in this chapter.

Problem Definition: Suppose that $\mathcal{C}$ is the collection of all classes in the DBpedia ontology DO and that $\mathcal{E}$ is a collection of entities extracted from a set of social media contents. Each entity $e \in \mathcal{E}$ is mapped to a class $c \in \mathcal{C}$ (i.e., $e \mapsto c$) according to the infobox information in Wikipedia. In this chapter, our aim is to (1) refine the classes of the entities (i.e., redo the classification such that $e \mapsto c'$, where $c' \in \mathcal{C}$) by performing class prediction using our proposed model of multi-level Naive Bayes Classifiers (mNBC), (2) generate hierarchical summaries $\mathcal{S}(\{E_1, E_2, ..., E_{|C|}\}, C, f(\cdot))$ using the entities with refined classes, and (3) find the most important entities, in terms of coherence, mapped to each class in each summary.

4.3 Refinement of Classes of Entities

In this section, we discuss how to refine the classes of entities so as to reduce the inaccuracy inherent in Wikipedia. To this end, we propose a lightweight yet effective classification model named multi-level Naive Bayes Classifiers. However, as our goal in this chapter is to generate summaries which consist of entities extracted from social media contents, we briefly introduce the extraction of entities first.

4.3.1 Extraction of Entities

Extraction of Entities is commonly know as “Entity linking” or “named entity disambiguation/normalization” in natural language processing. To detect entities in textual

documents, a knowledge base of entities is often required so that names in the documents can be linked to the entities of the knowledge base. A recently emerging direction is to use Wikipedia as the knowledge base and cross-link names to Wikipedia entities for entity extraction [43, 35, 60, 72], which is an instance of extremely fine-grained entity linking. The process of entity linking using Wikipedia is also called "wikification" [59]. DBpedia Spotlight [58] was proposed for annotating entities of DBpedia resources in text, which performs entity detection and name disambiguation to link unstructured information to the link data cloud via DBpedia. Despite various efforts made over the recent years, the problem remains challenging because of the inherent ambiguity in names/entities.

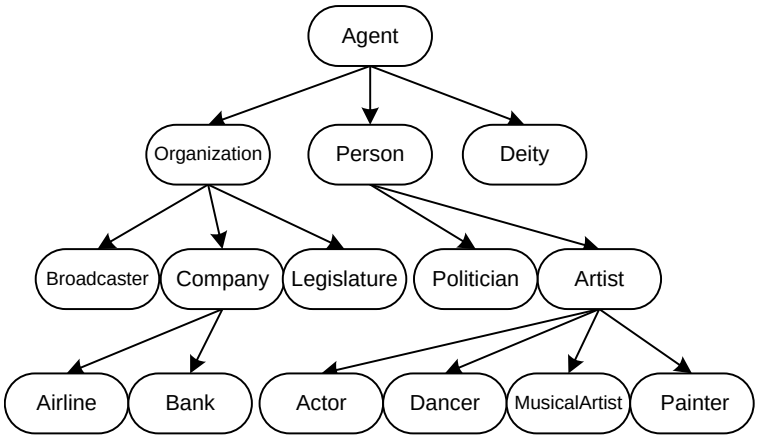
Since the extraction of entities stays beyond the focus of this work, we simply extract entities from social media contents by adopting the existing methods.

4.3.2 Multi-level Naive Bayes Classifiers

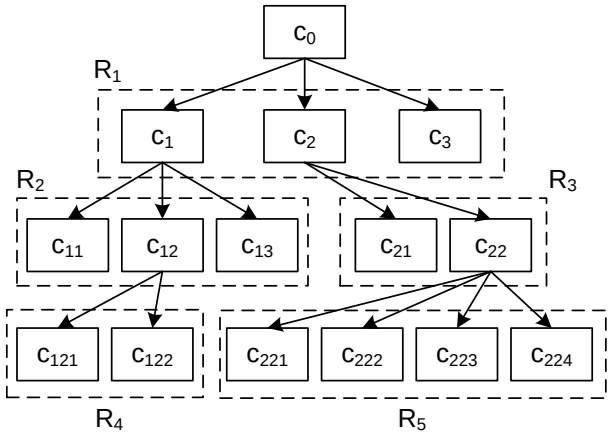
As mentioned earlier, Wikipedia infoboxes possess great inconsistency and inaccuracy because the creation and maintenance of the information are manually done by Wikipedia contributors, many of whom lack necessary knowledge and training on this. After all, it is quite challenging for ordinary contributors to choose the best from among thousands of infobox templates. For instance, a number of Wikipedia entities are labeled as "Person" instead of more specific classes such as "Artist", "Writer", "Philosopher". Since the inaccuracy of classes of entities would propagate to the summarization of social media contents and lead to inaccuracy of summaries more or less, it is better to refine the classes of entities to reduce the inaccuracy as much as possible.

To tackle this challenge, we propose a simple yet effective classification model named multi-level Naive Bayes Classifiers (mNBC). Although researchers have proposed various high-dimensional/hierarchical classification models [21, 52, 3, 38] and

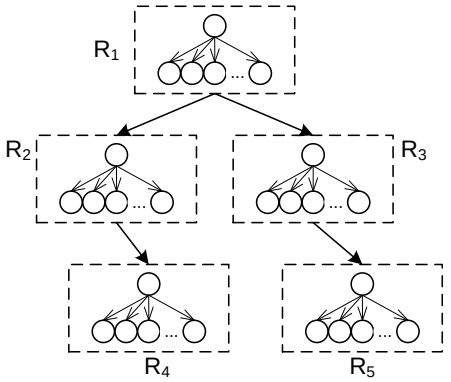
even other variants of hierarchical Naive Bayes Classifiers [75, 48], our model is more appropriate and especially designed to cater to the classification of Wikipedia entities with the aforementioned characteristics. Specifically, we build mNBC based on the DBpedia ontology by integrating both of them seamlessly. That is, a single Naive Bayes Classifier (NBC) is built for each set of classes sharing the same parent in the DBpedia ontology, and these individual NBCs are then integrated together to form an mNBC as a whole in accordance with the structure of the ontology.



(a) A simplified DBpedia ontology



(b) Formal representation



(c) mNBC

Figure 4.3: Structure of mNBC

Structure of mNBC

Figure 4.3 shows a typical structure of an mNBC model, where Figure 4.3(a) is a simplified DBpedia ontology for illustration purpose, Figure 4.3(b) is its formal representation and Figure 4.3(c) is the corresponding mNBC built for the ontology. In the formal representation of the ontology, boxes in solid line represent classes, which are connected by arrows. Classes sharing the same parent are put in one group and highlighted by a dashed box. We have five groups of classes in Figure 4.3(b), namely $R_1, R_2, ..., R_5$ respectively, for each of which an NBC is built, as shown in Figure 4.3(c). For each individual NBC, we assume that different properties are independent of each other given a class [36, 77].

In Figure 4.3(b), we use "c" plus a suffix to denote a class (except the root node) in the ontology in such a way that a suffix can reflect the level of a class and the relationship with the other classes: (1) The length of a suffix indicates at which level of the ontology a class is. For instance, c_2 is at the first level while c_{121} is at the third level. (2) If the suffix of class B has one more digit i at its end than that of class A, then B is the ith child of A. For instance, c_{122} is the second child of c_{12}, and c_{122} and c_{121} are siblings. Since the use of digits limits the number of children of a class to up to 10, in implementation we use the type "char" instead. Because each instance of it can represent at least 256 numbers in most programming languages while the maximum number of children a class has in the DBpedia ontology is less than 100. To avoid confusion, however, we will explicitly mention this suffix notation in the sequel only when we are using it. Otherwise, we do not determine the relationships among classes based on their suffixes. For instance, c_i and c_j can be arbitrary classes in the DBpedia ontology.

Modeling of mNBC

Suppose we have a set $\mathcal{E}$ of entities and each entity e has a set P of properties $\{p_1, p_2, ..., p_{|P|}\}$. The class of e is denoted by c_e. To avoid possible confusion between the class label of an entity and a class node in the DBpedia ontology, we use nod_c to denote the class node representing class c. However, as for a class node and its corresponding class, we use "node" or "class" interchangeably in the sequel if no ambiguity arises. For instance, it is equivalent to say that an entity is mapped to either a class node or a class in the DBpedia ontology. To build an mNBC model, each class node nod_c in the DBpedia ontology needs to maintain two statistics, one is the number of entities mapped to that class, denoted by $nod_c.cnt$, and the other is a property histogram, denoted by $nod_c.hist()$. Given any property p, $nod_c.hist(p)$ returns the number of entities that are mapped to nod_c and have property p.

Since entity e is classified as c_e, we can derive the entity's class chain according to the DBpedia ontology, so that both statistics of each class node in the class chain will be updated. Specifically, for every nod_c in the class chain, $nod_c.cnt$ and $nod_c.hist(p)$ should all be increased by 1 for every property $p \in P$.

Example 4.2. *Suppose that we have two entities e_1 and e_2, which are classified as c_{11} and c_{122} in Figure 4.3(b) respectively. e_1 has two properties $\{p_1, p_2\}$ while e_2 has one property $\{p_2\}$. According to the class labels of the two entities, the class chain for e_1 is c_1, c_{11} and that for e_2 is c_1, c_{12}, c_{122} (note again that c_0 is not considered as a class in this work). Thus given e_1 we update the statistics of class node nod_{c_1} and $nod_{c_{11}}$, and given e_2 we update those of class node nod_{c_1}, $nod_{c_{12}}$ and $nod_{c_{122}}$. After updating, we have (1) $nod_{c_1}.cnt = 2$, (2) $nod_{c_{11}}.cnt = nod_{c_{12}}.cnt = nod_{c_{122}}.cnt = 1$, (3) $nod_{c_1}.hist(p_1) = nod_{c_{11}}.hist(p_1) = nod_{c_{11}}.hist(p_2) = nod_{c_{12}}.hist(p_2) = nod_{c_{122}}.hist(p_2) = 1$, and (4) $nod_{c_1}.hist(p_2) = 2$. Note that $nod_{c_1}.cnt = 2$ because both e_1 and e_2 have c_1 in their class chain, and $nod_{c_1}.hist(p_2) = 2$ because both e_1 and e_2 have property p_2.*

By updating the statistics of all the class nodes in the DBpedia ontology using entities in the training data set, we actually have already finished building the mNBC model. The ontology with updated statistics will serve as the mNBC model for classification.

Classfication of Entities Using mNBC

Recall that we mentioned earlier that an NBC would be built for every set C of classes sharing the same parent. Suppose we already have such an NBC which will be used to classify an entity e with properties $p_1, p_2, ..., p_{|P|}$, where $|P|$ is the number of properties e has. Let $x = (p_1, p_2, ..., p_{|P|})$ be the property vector of e. The probability for e to be classified as c given x is calculated as follows.

$$
\begin{aligned}
p(c|x) &= \frac{p(x|c) \cdot p(c)}{p(x)} && (4.1)\\
&= \frac{\prod_{i=1}^{|P|} p(x_i|c) \cdot p(c)}{\cancel{p(x)}} && (4.2)\\
&\propto \prod_{i=1}^{|P|} p(x_i|c) \cdot p(c) && (4.3)
\end{aligned}
$$

Formulae 4.1 to 4.3 are standard solutions commonly used for an NBC model [77]. Formula 4.1 applies Bayes' theorem to $p(c|x)$. In Formula 4.2, $p(x)$ is omitted because it is a constant, so that $p(c|x)$ can be written in a proportional form as in Formula 4.3. Next, we rewrite the probabilities in Formula 4.3 as frequencies to further simplify the calculation.

$$p(c|x) \propto \prod_{i=1}^{|P|} \frac{freq(x_i|c)}{freq(c)} \cdot \frac{freq(c)}{\sum_j freq(c_j)} \tag{4.4}$$

$$= \frac{\prod_{i=1}^{|P|} freq(x_i|c)}{freq(c)^{|P|-1}} \cdot \frac{1}{\cancel{\sum_j freq(c_j)}} \tag{4.5}$$

$$\propto \frac{\prod_{i=1}^{|P|} freq(x_i|c)}{freq(c)^{|P|-1}} \tag{4.6}$$

In Formula 4.4, $p(x_i|c)$ is written as $freq(x_i|c)/freq(c)$, where $freq(x_i|c)$ is the number of entities having a certain property x_i (i.e., p_i) given that those entities are classified as c and $freq(c)$ is the number of entities classified as c. Besides, $p(c)$ is written as $freq(c)/\sum_j freq(c_j)$, where $freq(c_j)$ is the number of entities classified as a certain class $c_j \in C$. Note that in these formulae c is a random variable while c_j is one certain class in C. A factor $freq(c)$ in the numerator is canceled with one same factor in the denominator, so that Formula 4.4 is rewritten as Formula 4.5. Since $\sum_j freq(c_j)$ is a constant with regard to any certain NBC, this factor is omitted and $p(c|x)$ is finally written as the one in Formula 4.6.

Recall that we maintain two statistics for each class node in the DBpedia ontology. For any class node nod_c and property x_i, we have $freq(x_i|c) = nod_c.hist(x_i)$ and $freq(c) = nod_c.cnt$. Since $nod_c.hist(x_i)$ and $nod_c.cnt$ are already known, we can calculate the value of Formula 4.6 easily. Note that $p(c|x)$ does not equal the value of Formula 4.6 because we have omitted two constant factors in Formula 4.2 and 4.5. Therefore, we first calculate the value of Formula 4.6 for every $c \in C$ and then normalize them to obtain the value of $p(c|x)$. Although it is easy to find out the maximum value of Formula 4.6 for all classes in C to determine the class of e, performing normalization is still needed so as to terminate the classification process properly, which will be discussed later.

Through the above calculations, we can classify an entity e as a class c with the highest $p(c|x)$ in a single NBC model. After building an NBC for each set of classes with the same parent in the DBpedia ontology, we build a collection of NBCs which altogether form a model of multi-level Naive Bayes Classifiers, i.e., an mNBC as shown in Figure 4.3(c). In mNBC, the set of classes are different for each individual NBC, and the distribution of properties varies accordingly, which is reflected by the property histogram of each class node.

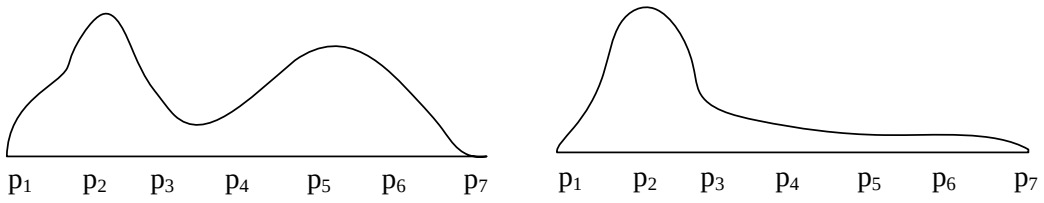
Given an entity, we can classify it in an iterative manner so as to predict and refine the class of the entity level by level, from top to bottom in the mNBC model. The resultant predicted classes at different levels then form the class chain of the entity.

Example 4.3. *Given an entity e and its property vector x, we illustrate the classification process based on Figure 4.3. We start the classification from the top NBC highlighted by dashed box R_1 by applying Formula 4.6 three times to find the largest value. If $p(c_3|x)$ is the largest, we classify e as c_3 and terminate the classification because c_3 has no descendants in the ontology. However, if $p(c_1|x)$ is the largest, we then classify e as c_1 at the first level, and come to the second level to classify e using the NBC highlighted by dashed box R_2 because R_2 corresponds to the set of children of c_1. Suppose this time e is classified as c_{12}. Since c_{12} still has children, we continue to classify e using the NBC highlighted by dashed box R_4. Suppose e is finally classified as c_{122}, then we say that the class of e is c_{122} with c_1, c_{12}, c_{122} as its class chain.*

Zero Frequency Correction

Consider Formula 4.6 again. If, in the training data set, all the entities mapped to class node nod_c have no property x_i, $freq(x_i|c)$ becomes 0. For an entity from the testing data set which has this property, the situation will lead to $p(c|x) = 0$, even if $p(c|x)$ might be quite large if property x_i is not considered for the calculation. In

this case, a zero $freq(x_i|c)$ will cancel the effect of all the other non-zero factors in Formula 4.6. To avoid this problem, we can perform zero frequency correction, which is inspired by the Laplacian correction [34], whenever $freq(x_i|c)$ equals 0. Next we use an example to briefly illustrate how the correction is performed.



(a) nod_{c1}.hist (b) nod_{c2}.hist

Figure 4.4: Histograms of properties for two class nodes

Example 4.4. *Figure 4.4 draws two curves reflecting the histograms of properties for two class nodes nod_{c_1} and nod_{c_2}. The height of the curves indicates approximately the distribution of entities in terms of properties. Suppose $nod_{c_1}.cnt = nod_{c_2}.cnt = 10$ (i.e., 10 entities are mapped to class c_1 and c_2 respectively). For c_1, we have $nod_{c_1}.hist(p_2) = 9$, $nod_{c_1}.hist(p_5) = 8$ and $nod_{c_1}.hist(p_7) = 0$. For c_2, we have $nod_{c_2}.hist(p_2) = 10$, $nod_{c_2}.hist(p_5) = 2$ and $nod_{c_2}.hist(p_7) = 1$. Given an entity e with property vector $x = (p_2, p_5, p_7)$, we calculate the unnormalized probability according to Formula 4.6 for each class as follows: $p(c_1|x) \propto (9 \times 8 \times 0)/10^{(3-1)} = 0$ and $p(c_2|x) \propto (10 \times 2 \times 1)/10^{(3-1)} = 0.2$. Since one probability is 0, we cannot normalize the two values. But we can simply classify e as c_2 because the value for c_2 is larger. By comparing the three properties of entity e against the two curves in Figure 4.4, however, we notice that intuitively e is more likely to be classified as c_1 instead, because most entities mapped to c_1 have both property p_2 and p_5 while most entities mapped to c_2 only have property p_2. To avoid this misclassification, we assume that a new entity having all the seven properties are added to the training data set to update the statistics of nod_{c_1}. To achieve this, we simply increase each factor in Formula 4.6 by 1 on the fly as long as any factor*

is 0. As a result, we have $p(c_1|x) \propto ((9+1)\times(8+1)\times(0+1))/(10+1)^{(3-1)} \approx 0.74$. *Now since 0.74 is larger than 0.2, entity* e *is classified as* c_1.

In reality, the number of entities mapped to a class node is so large that we can omit the tiny effect introduced by performing the zero frequency correction.

Stop Condition

Recall that in Example 4.3, we classify entity e as c_{122} by following the hierarchical structure of the ontology in Figure 4.3 from the first level to the last. This is not often the case, however. We may have to terminate the recursive process after classifying e as c_1 and c_{12}, because the probability for e to be classified as c_{122} may not be large enough. Given a minimum probability threshold p_{min}, to determine whether the maximum probability $p(c|x)$ is larger than p_{min}, normalization should be performed among the values of Formula 4.6 for all $c \in C$.

In spite of normalization, we cannot rule out all the exceptions. This is because the probability $p(c|x)$ for a certain class c might be incorrectly scaled up to a large value owing to the small probabilities of other classes during the normalization. Take an extreme case as an example. If only one class c has non-zero unnormalized value according to Formula 4.6, the normalization will lead to the probability of that class being 1. This may not be true if there is only a small overlap between the properties of entity e and those in the histogram of class node nod_c. Therefore, to further validate a classification, we multiply $p(c|x)$ by a property proportion $prop$ to obtain the revised probability $p_{rev}(c|x)$, which is used as the final probability to compare with p_{min}. The formula is written as follows.

$$p_{rev}(c|x) = p(c|x) \cdot prop \tag{4.7}$$

$$= p(c|x) \cdot \frac{|P \cap pp(nod_c.hist)|}{|P|} \tag{4.8}$$

where P is the set of properties of entity e, $|P|$ is the cardinality of P and $pp(nod_c.hist)$ is the set of properties in the histogram of node nod_c. Therefore, the property proportion $prop$ is the percentage of the common properties shared by entity e and node nod_c with regard to all properties of e. Only if $p_{rev}(c|x)$ is not less than p_{min}, we consider that entity e is classified as c. With this stop condition, we give the algorithm of classifying an entity using mNBC below.

Algorithm 3: The mNBC Algorithm

Input: the property vector $x = (p_1, p_2, ..., p_{|P|})$ of entity e (P is the property set of e), the DBpedia ontology DO with updated statistics, the probability threshold p_{min}

Output: a class chain L of entity e

```
1  begin
2      initialize C to be the set of classes at level 1 of DO;
3      initialize L to be an empty list;
4      while true do
5          compute values of Formula 4.6 for all c ∈ C;
6          normalize the values to obtain p(c|x) for all c ∈ C;
7          let c_tmp = arg max_{c∈C} p(c|x);
8          calculate p_rev(c_tmp|x) according to Formula 4.8;
9          if p_rev(c_tmp|x) < p_min then
10             return L;
11         append c_tmp to the end of L;
12         let C be the set of children of c;
13         if C = ∅ then
14             return L;
15 end
```

Algorithm 3 performs a greedy search in the DBpedia ontology and returns the class

chain for an input entity, in which the last element is the class of the entity. It starts from the classes at the first level of the ontology (line 1 to 4), finds a class c_{tmp} with the maximum probability $p(c|x)$ and calculates its revised probability $p_{rev}(c|x)$ (line 5 to 8). If this probability is less than the threshold p_{min}, the algorithm terminates (line 9 to 10). Otherwise, c_{tmp} is appended to a list storing the class chain of the entity (line 11), and the algorithm again starts to examine the children of c_{tmp} (line 12) in a recursive manner until there is no child for c_{tmp} (line 13 to 14).

4.4 Summarization

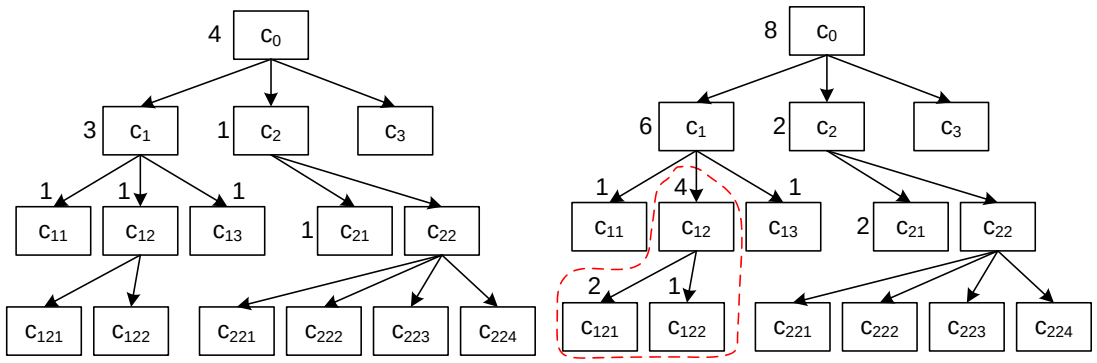
In last section, we discussed how to build the mNBC model and refine the class labels of entities based on this model. Next, we introduce how to generate hierarchical summaries based on the DBpedia ontology given entities with refined classes. Firstly, we map those entities onto the DBpedia ontology. Then we split the ontology into multiple sub-hierarchies, each of which can be perceived as a summary. Considering the fact that too many entities may be mapped to one class node, we further propose to rank the entities so as to select the most important ones.

4.4.1 Entity Mapping

Mapping entities onto the DBpedia ontology can be done at the same time when doing the classification. As described in Algorithm 3, we classify an entity e along the hierarchical structure of the DBpedia ontology from top to bottom, until the stop condition is satisfied. We can remember the last valid class node nod_c which is appended at the end of the class chain during the recursive process as the algorithm executes. At the time of termination, nod_c will be the very node to which entity e should be mapped. In the sequel, when we say that an entity e is mapped to a class node nod_c, the class

node refers to the one corresponding to the last class c at the end of the class chain of the entity unless otherwise stated. That is, we map an entity e to its lowest class node nod_c in the ontology when generating summaries because c is the most specific class to describe e, although e can also be classified as those classes corresponding to the ancestor nodes of nod_c.

To simulate the mapping operation, another two statistics of nod_c need to be updated, which are $nod_c.ent\_cnt$ and $nod_c.ent\_hist$ respectively. Note not to confuse these two statistics with $nod_c.cnt$ and $nod_c.hist$ used for building mNBC. $nod_c.ent\_cnt$ stores the number of entities mapped to class node nod_c as well as those mapped to the descendants of nod_c, i.e., the total number of entities mapped to the sub-hierarchy rooted at nod_c. $nod_c.ent\_hist$ stores the histogram of entities only mapped to nod_c, where $nod_c.ent\_hist(e)$ returns the frequency that entity e is mapped to nod_c. Thus when entity e is finally mapped to class node nod_c when Algorithm 3 terminates, we update $nod_c.ent\_cnt$ and $nod_c.ent\_hist(e)$ by increasing them by 1. We should also increase $nod_{c_{anc}}.ent\_cnt$ by 1 for each $c_{anc} \in ancestors(c)$ where $ancestors(c)$ is the set of ancestor nodes of c. This is because the statistic ent_cnt of a node needs to store the total number of entities mapped to the sub-hierarchy rooted at that node.



(a) Four entities mapped to the ontology

(b) Eight entities mapped to the ontology ($4 \neq 2+1$ because one entity is mapped to $nod_{c_{12}}$)

Figure 4.5: Entity mapping

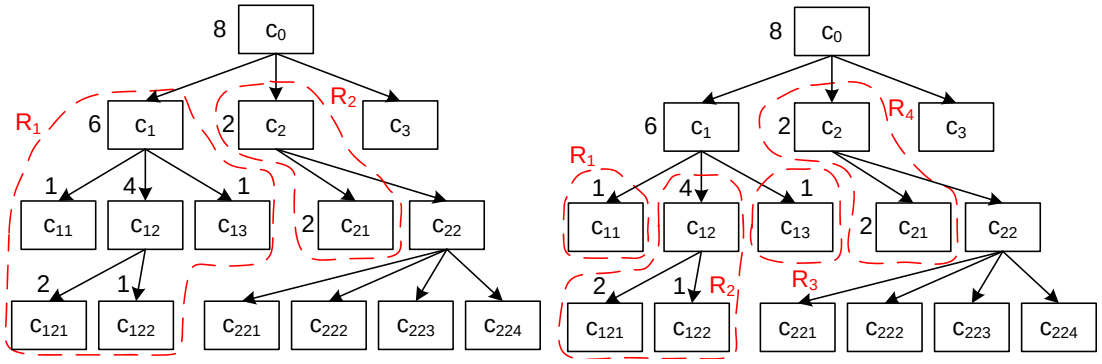
Figure 4.5 demonstrates the mapping of entities to the ontology shown in Figure 4.3. In Figure 4.5(a), four entities are mapped to four different class nodes, i.e., $nod_{c_{11}}$, $nod_{c_{12}}$, $nod_{c_{13}}$ and $nod_{c_{21}}$. The number next to a class node indicates how many entities are mapped to the sub-hierarchy rooted at that class node, while there is no number if no entity is mapped. For instance, the number 1 next to $nod_{c_{12}}$ means $nod_{c_{12}}.ent\_cnt = 1$, i.e., only one entity is mapped to the sub-hierarchy rooted at $nod_{c_{12}}$. Since the sub-hierarchy has three class nodes $nod_{c_{12}}$, $nod_{c_{121}}$ and $nod_{c_{122}}$, and since there is no entity mapped to the other two class nodes, we know that the single entity must be mapped to $nod_{c_{12}}$. The number 3 next to nod_{c_1} means $nod_{c_1}.ent\_cnt = 3$, i.e., three entities are mapped to the sub-hierarchy rooted at nod_{c_1}. Figure 4.5(b) shows the ontology after four more entities are mapped. $nod_c.ent\_cnt$ is also updated accordingly for any $c \in C$, where C is the set of classes in the ontology. Note that in the sub-hierarchy highlighted by the dashed line in Figure 4.5(b), we have $nod_{c_{12}}.ent\_cnt = 4$, $nod_{c_{121}}.ent\_cnt = 2$ and $nod_{c_{122}}.ent\_cnt = 1$. $4 \neq 2{+}1$ because an entity e is actually mapped to $nod_{c_{12}}$. The number of entities mapped to the sub-hierarchy rooted at a class node equals the sum of the number of entities mapped to that class node and the number of entities mapped to the sub-hierarchies rooted at the children of that class node. This relationship of statistics among a class node and its children is captured by the formula as follows.

$$\begin{aligned} nod_c.ent\_cnt &= \sum_{e_i \in E} nod_c.ent\_hist(e_i) \\ &+ \sum_{c_j \in children(c)} nod_{c_j}.ent\_cnt \end{aligned} \tag{4.9}$$

where E is the set of entities mapped to class node c and $children(c)$ is the set of children of c.

4.4.2 Summary Generation

Once a set of entities have been mapped onto the DBpedia ontology, hierarchical summaries can be generated naturally and easily by cutting off sub-hierarchies from the ontology. Given a threshold δ which sets an upper bound for the number of entities allowed to contain in each summary, the ontology is traversed to find a set C_{root} of classes. For any $c \in C_{root}$, $nod_c.ent\_cnt$ is less than or equal to δ while $nod_{parent(c)}.ent\_cnt$ ($parent(c)$ is the parent of c) is larger than δ if the parent of c exists. In this case, each sub-hierarchy rooted at a class in C_{root} is then a hierarchical summary. Depth-first search (DFS) can be performed for the traversal of the ontology. Recall Definition 4.2 that a summary $\mathcal{S}(\{E_1, E_2, ..., E_{|C|}\}, C, f(\cdot))$ consists of a set of classes, subsets of entities corresponding the classes and an injective function mapping the entity sets to the classes. The entities constitute the content of the summary while the classes and the function reflect the organization and structure of the summary.



(a) Two summaries when $\delta = 6$

(b) Four summaries when $\delta = 4$

Figure 4.6: Summary generation

Figure 4.6 demonstrates the generation of summaries given different values of threshold δ. In Figure 4.6(a) with δ set to 6, we need to find the "uppermost" classes so that the sub-hierarchies rooted at these classes contains at most 6 entities. A class c being uppermost guarantees that $nod_c.ent\_cnt$ is less than or equal to the threshold while that

of its parent is not. Apparently, c_1 and c_2 satisfy the requirement. Therefore, we obtain two sub-hierarchies, rooted at c_1 and c_2 respectively, which are the summaries when δ is 6. The two summaries are highlighted in dashed line in Figure 4.6(a). Note that since no entity is mapped to c_{22} and its descendants, we omit and remove them from the sub-hierarchy rooted at c_2, to make the corresponding summary (labeled R_2) more compact. When δ is set to 4, the resultant summaries are shown and highlighted in Figure 4.6(b). Now there are four summaries in the ontology, three of which is generated by splitting the one labeled R_1 in Figure 4.6(a). Intuitively, decreasing the threshold δ will push the uppermost class of a summary to lower levels and even split a large summary into several smaller ones, thereby resulting in more or smaller summaries, which tend to be more specific and coherent in terms of entities contained. In a word, the threshold δ adjusts the granularity of generated summaries.

4.4.3 Top Entities Selection

The distribution of entities extracted from different sets of social media contents often varies greatly. Entities in certain type of classes (e.g., "City" and "Artist") are more frequently mentioned, so that hundreds of entities or even more can be mapped to one single class. To better distinguish among so many entities, we next propose a ranking procedure in terms of coherence so that the most important entities can be ranked higher than others. Before presenting the details, we introduce two new concepts first.

Definition 4.3. *Given two classes in the DBpedia ontology, we denote by* **comm** *the number of their common ancestor classes (inclusive of the classes themselves) and by* **dist** *the minimum number of hops (or edges) from one class to reach the other.*

Example 4.5. *Take the ontology in Figure 4.6 as an example. For classes* c_1 *and* c_{122}*,* $comm$ *is 1 because only* c_1 *is their common class while* $dist$ *is 2 because there are*

two hops from c_1 to c_{122} (i.e., $c_1 \rightarrow c_{12}$ and $c_{12} \rightarrow c_{122}$). For c_{121} and c_{13}, $comm$ is still 1 because only c_1 is their common class while $dist$ is 3 because there are three hops from c_{121} to c_{13} (i.e., $c_{121} \rightarrow c_{12}$, $c_{12} \rightarrow c_1$ and $c_1 \rightarrow c_{13}$). For another pair of classes c_{121} and c_3, $comm$ is 0 because there is no common class for them (recall that c_0 is not considered as a class in this work) while $dist$ is 4 because of four hops from c_{121} to c_3 (i.e., $c_{121} \rightarrow c_{12}$, $c_{12} \rightarrow c_1$, $c_1 \rightarrow c_0$ and $c_0 \rightarrow c_3$). Note that c_0 should be considered as a node when calculating $hist$, otherwise the path between two classes will be disconnected.

The variable $comm$ measures the closeness between two classes in the DBpedia ontology in terms of the number of common ancestors while $dist$ measures the distance of the classes. By combining these two variables, we introduce a new metric below to reflect the compactness and coherence between two classes.

$$coh(c_i, c_j) = \frac{comm_{ij}}{dist_{ij}} \tag{4.10}$$

where $comm_{ij}$ and $dist_{ij}$ are the values of $comm$ and $dist$ given two classes c_i and c_j. The larger $comm_{ij}$ is (or the smaller $dist_{ij}$ is), the more coherent two classes c_i and c_j are in the ontology. To calculate $coh(c_i, c_j)$ efficiently, we propose to obtain $comm_{ij}$ and $dist_{ij}$ by making use of the suffix notation of classes, which has been neatly designed. We demonstrate the calculations using the following example.

Example 4.6. *Recall that in Section 4.3.2 we design to use the suffix of a class to reflect the position of the class and its relationship with other classes. For instance, c_{121} is at the third level of the ontology because the length of the suffix is 3 while c_{12} is the parent of c_{121} because c_{12} is at the second level and "12" is a prefix of "123". In this case, given any two classes, $comm$ equals the length of the common prefix shared by their*

suffixes, and $dist$ *equals the sum of the lengths of the rest suffixes. For instance, given* c_{121} *and* c_{13}*,* $comm$ *is 1 because their suffixes share a 1-length common prefix "1", and* $dist$ *is 3 because the remainders of the two suffixes after the removal of the common prefix are "21" of length 2 and "3" of length 1 respectively. Note that the common prefix must start from the very beginning of both suffixes. For instance, although* c_{121} *and* c_{21} *have a common substring "21" in their suffixes, "21" is not the prefix of* c_{121}*, thereby leading to* $comm = 0$ *and* $dist = 3 + 2 = 5$*.*

Given two entities, we can evaluate the coherence of them by calculating $coh(c_i, c_j)$ if the entities are mapped to c_i and c_j respectively. Since our goal is to rank entities mapped to a certain class c, next we introduce a formula to score these entities based on Formula 4.10.

$$score(e_i) = \log(nod_c.ent\_hist(e_i) + 1) \cdot \sum_{j=1}^{n} \frac{comm_{ij} + 1}{dist_{ij} + 1} \tag{4.11}$$

where e_i is the ith entity mapped to class c and n is the number of entities which are extracted from the set of social media contents having e_i and are contained in the same summary as e_i. This formula consists of two factors. The first factor is a logarithm expression, the value of which is larger if e_i appears in more social media contents. An extra 1 is added to the frequency of e_i to ensure that the whole factor is larger than 0. The second factor is a revised version of Formula 4.10 by adding 1 to both $comm_{ij}$ and $dist_{ij}$ of each fractional addend to avoid the numerator or denominator being 0. It will be larger if more entities extracted from social media contents having e_i are also mapped into the same summary. In other words, entity e_i will be scored higher if it coexists in a summary with more entities from various social media contents. Note that the n entities involved for the calculation of the second factor are those in the same

set of social media contents and summary as e_i, including e_i itself. We also define that $comm_{ij} = dist_{ij} = 0$ if $i = j$. Thus when $n = 1$ (i.e., only one entity e_i is extracted from the set of social media contents having e_i, which is an extreme case and very rare), the second factor in Formula 4.11 becomes 1 and Formula 4.11 is reduced to $score(e_i) = \log(nod_c.ent\_hist(e_i) + 1)$. If $n > 1$, the second factor will be greater than 1. Since the first factor is always greater than 0 and the second one is always greater than or equal to 1, $score(e_i)$ will always be positive, and be greater than or equal to the first factor.

4.5 Experimental Study

4.5.1 Data Sets

In this section, we exploit the data sets in the English version of DBpedia 3.9 for the experimental study. There are 529 classes in total described by 2,333 different properties. The English version contains 3.22 million entities which are classified and mapped onto the DBpedia ontology, including 832,000 persons, 639,000 places, 372,000 creative works, 226,000 species, 209,000 organizations and so on[10]. Specifically, three data sets in DBpedia 3.9 are used, which are "DBpedia Ontology", "Mapping-based Types" and "Mapping-based Properties" respectively. The first data set, in the OWL[11] (Web Ontology Language) format, describes the parent-child relationships between classes, from which the hierarchical structure of the DBpedia ontology can be constructed. The other two, in the Turtle[12] format, record the entity-classes mappings and entity-properties relationships respectively, which are extracted from corresponding English Wikipedia articles/infoboxes. These two data sets are mapping-based in that the data is extracted

[10]http://wiki.dbpedia.org/Ontology39
[11]http://www.w3.org/2001/sw/wiki/OWL
[12]http://www.w3.org/TR/turtle/

based on hand-generated mappings of Wikipedia infoboxes/templates to the DBpedia ontology[13]. This standardizes the extracted classes and properties to avoid cases that different infoboxes are used for the same class or different property names are used for the same property. Note that in the "Mapping-based Types" data set, an entity may have multiple classes, which is similar to the concept of class chain in this chapter.

To extract entities, we crawl various sets of tweets, i.e., the type of social media contents published on Twitter, by setting different query strings based on Twitter's REST APIs[14]. All the tweets in each set contains a certain query string, such as "Singapore", "White House", "Barack Obama", to make sure that the tweets are all related to that query string. For instance, we have crawled one set of tweets by specifying the query string as "Singapore", with around 1.867 million tweets crawled during a period of two months. By default, we use this set of tweets to extract entities in this section unless otherwise stated.

4.5.2 Evaluation of mNBC

The aim of proposing the mNBC model is to help refine the classes of entities so as to generate high-quality summaries. In addition, the model can also be used to perform class prediction for new entities as long as a set of properties are given. Therefore, a high performance is vital for mNBC to achieve these objectives. Next, we conduct a series of experiments to evaluate the performance of mNBC.

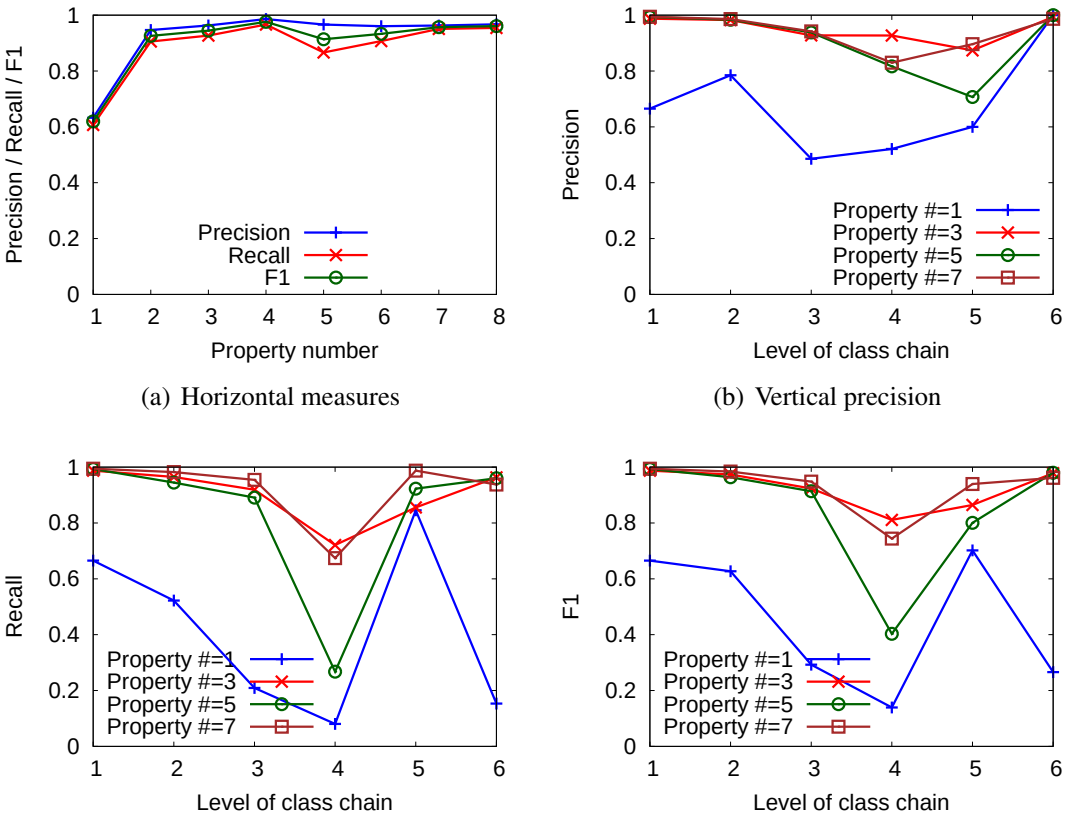
Precision and recall are two measures commonly used to evaluate models in pattern recognition and information retrieval. However, they are usually adopted for the evaluation of binary classifications. In our scenario, an entity e can be classified as multiple classes in the class chain $c_i, c_{i+1}, ..., c_{j-1}, c_j$. Although the last class c_j in the class chain

[13]http://wiki.dbpedia.org/Datasets
[14]https://dev.twitter.com/rest/public

Table 4.1: Entities and their corresponding class chains

Entity	Class Chain
e_1	$c_{11}, c_{12}, c_{13}, c_{14}$
e_2	$c_{21}, c_{22}, c_{23}, c_{24}, c_{25}, c_{26}$
e_3	c_{31}, c_{32}, c_{33}

is the best and most specific class for e, we cannot say that it is completely wrong when e is classified as some other class in the class chain. Suppose that Table 4.1 displays a few entities and their class chains generated by Algorithm 3. Recall that each entity in the training data set "Mapping-based Types" also corresponds to multiple classes. We organize these classes for each entity in order according to the levels they are in the DBpedia ontology to produce a training class chain for each entity, so that we can make comparisons between the training class chains and those generated by our mNBC model level by level. For an entity, we divide the number of true positive classes by the number of classes in the class chain generated by mNBC to obtain the precision while divide that by the number of classes in the training class chain to obtain the recall. We call the two measures the **horizontal precision/recall** in the sequel because they are calculated for every single entity in a certain row as shown in Table 4.1. Likewise, for each level of the class chains, we divide the number of true positive classes at that level by the number of classes in the class chains generated by mNBC at that level to obtain the **vertical precision** while divide that by the number of classes in the training class chains at that level to obtain the **vertical recall**. The horizontal precision and recall evaluate our model in terms of each entity whereas the vertical ones evaluate it in terms of each level of the class chains across all the entities. We perform 10-fold cross validation to avoid overfitting in the evaluation. Since the horizontal measures are calculated for each entity, we average them to obtain the final horizontal precision/recall.



(a) Horizontal measures

(b) Vertical precision

(c) Vertical recall

(d) Vertical F1 score

Figure 4.7: Precision, recall and F1 score for different number of properties

Effect of Different Number of Properties

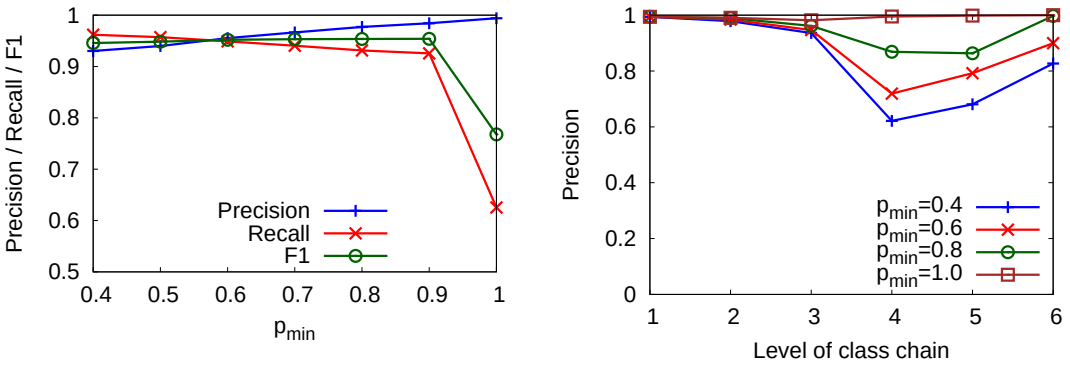
Figure 4.7 shows the variation of horizontal and vertical precision/recall/F1 score[15] with regard to entities with different number of properties. That is, the training entities are put into different groups according to how many properties they have and the horizontal/vertical measures are calculated for each group. p_{min} is set to 0.7. In Figure 4.7(a), horizontal measures are presented when the number of properties ranges from 1 to 8. The three measures are relatively small when property number is 1. However, when entities have more properties (from 2 to 8), the three measures increase significantly, with the values larger than 0.9 in most cases. This discovery indicates that

[15] $F1 = 2 \cdot precision \cdot recall/(precision + recall)$

in order to obtain good classification result, more properties is needed for each entity. Figure 4.7(b), 4.7(c) and 4.7(d) also confirm this indication by presenting the vertical precision, recall and F1 score at different levels of class chains. The blue line represents the measures when entities only have one property, which is beneath all the other three lines representing measures when entities have more properties. With more properties, the three measures are quite large at the first two levels of the class chains while the fluctuation becomes obvious for the rest levels. This is because entities are more distinguishable in terms of classes at top levels of the DBpedia ontology. For instance, given an entity Camel with five properties "phylum, order, kingdom, family, class", it is easy to classify it as "Species" rather than "Place" ("Species" and "Place" are classes at the first level in the DBpedia ontology) while it is challenging when it comes to "Mammal" and "Fish" (which are classes at the fourth level). However, we obtain high horizontal and vertical measures for entities with more properties overall.

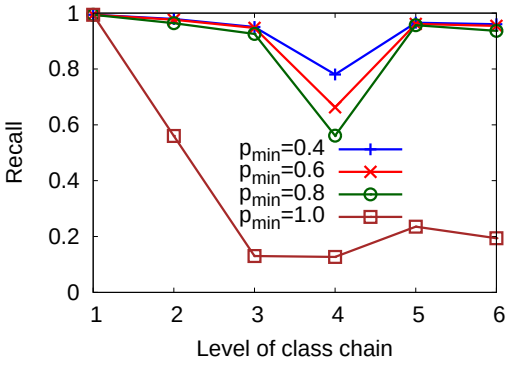
Effect of Different Values of p_{min}

Figure 4.8 shows the horizontal/vertical measures with regard to different values of the probability threshold p_{min}. The larger p_{min} is, the more conservative the mNBC model is, the earlier Algorithm 3 terminates. We use entities with at least three properties. Figure 4.8(a) presents the horizontal measures as p_{min} ranges from 0.4 to 1 with 0.1 as the step value. The horizontal precision always keeps increasing linearly as p_{min} increases. The horizontal recall decreases also linearly but slowly as p_{min} increases from 0.4 to 0.9, leading to F1 score almost remaining unchanged during the process. Except for the sudden drop of recall and F1 score when $p_{min} = 1$, the three measures stay with high values for most cases. The sudden drop of recall is because the classification process (i.e., Algorithm 3) is terminated so early that the class chains generated by mNBC tend to be shorter than the training class chains. With the lower bound probability set to

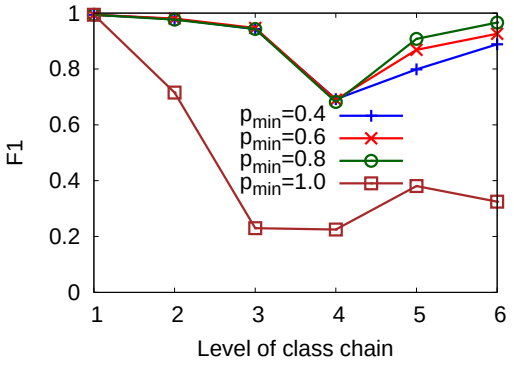


(a) Horizontal measures

(b) Vertical precision



(c) Vertical recall



(d) Vertical F1 score

Figure 4.8: Precision, recall and F1 score for different p_{min}

1, entities will not be classified as classes at lower levels of the DBpedia ontology, even if the probability reaches as high as 0.99. As a result, p_{min} set to 1 should be avoided in Algorithm 3.

Figure 4.8(b) to 4.8(d) show the vertical measures separately given different values of p_{min} with regard to different levels of class chains. The vertical precision keeps increasing as p_{min} increases in Figure 4.8(b) while the vertical recall decreases gradually at the same time in Figure 4.8(c), leading to the F1 score not changing too much except for the case when $p_{min} = 1$ as shown in 4.8(d). Similar to Figure 4.8(a), the values of vertical recall and F1 score are small when $p_{min} = 1$. Again the fluctuation is observed when it comes to lower level classes. The vertical measures drop obviously at the fourth

level in both Figure 4.7 and 4.8, probably because this level of the DBpedia ontology has the largest number of classes, i.e., 195 out of 529. Note that sometimes the decrease of the measures does not necessarily mean misclassification because the mNBC model is capable of making refinement, which will be falsely regarded as misclassification by using the training class chains as gold standard. Later, we will study how much improvement can be made through the refinement of mNBC.

Distribution of Probabilities

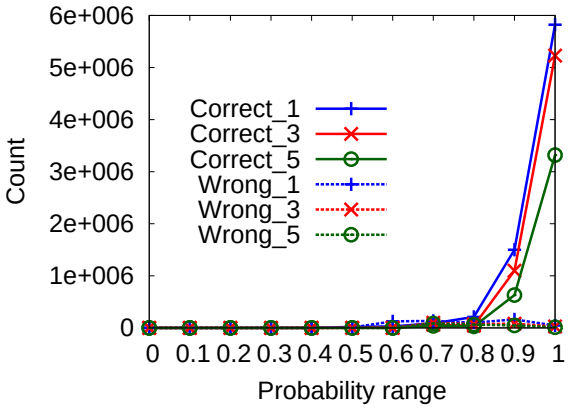


Figure 4.9: Distribution of probabilities

Algorithm 3 calculates a probability for every class in the class chain. We set p_{min} to 0.7 and investigate the distribution of probabilities of the classes in each class chain generated by mNBC, which is depicted in Figure 4.9. Note that the probability of a class can be smaller than p_{min} here as long as it is not less than the probability of the previous class in a class chain. That is, the probabilities of the classes in a class chain monotonically increase from the first class to the last one. Each of the numbers along the x-axis denotes a probability range from that number (inclusive) to the closest number (exclusive) on the right. For instance, 0 denotes the range $[0, 0.1)$ and 0.9 denotes $[0.9, 1)$. However, 1 denotes the case when the probability is 1. The y-axis indicates

the number of classes falling into certain probability ranges. In the notations of the six lines in Figure 4.9, "Correct" and "Wrong" indicate whether entities are classified correctly or not, compared with the training class chains, while the numbers appended to "Correct" and "Wrong" indicate the minimum number of properties entities have. For instance, "Correct_3" shows the distribution of probabilities when entities having at least 3 properties are correctly classified. We notice that the three solid lines have large values for the probability ranges from 0.9 to 1 while they have quite small values for the rest ranges. In addition, the three dashed lines also have small values for almost all the ranges. Although we allow the probabilities in a class chain to increase monotonically from a value smaller than p_{min}, which is less strict than the condition in Algorithm 3, we still notice that all the six lines have quite small values for the probability ranges from 0 to 0.8. In a word, most entities are correctly classified with high probability by the mNBC model.

Improvement by mNBC

Previously we calculated the horizontal/vertical measures by considering the training class chains as the gold standard. However, these measures cannot reflect the capability of improvement or refinement made by the mNBC model. Besides, evaluating the improvement in classification is also hard. Therefore, we next randomly select some sample entities and try to manually evaluate the improvement made by mNBC. Specifically, we randomly select three sets of entities, with 300, 400 and 500 entities for every set, where each entity has at least three properties. Then we generate class chains for these entities using our mNBC model, and compare them with the training class chains manually. We try to figure out the differences between the two group of class chains for each entity and see whether our model can make any refinement. We set p_{min} to 0.5. As shown in Table 4.2, there are 58, 78 and 98 entities respectively, in the three sets,

the generated class chains of which are different from the corresponding training class chains. Among these entities, the class chains of 20 to 30 entities have been obviously refined by our model after careful examination. The refine rate, calculated by dividing the number of improved entities by that of different ones, shows that around 1/3 "disputable" entities can be classified as a better or more proper class by using our model. The overall accuracy is as high as $86\% - 87.3\%$, which is the ratio of the number of correctly classified entities (including those with the same class chains in both groups and those that have been refined) to that of all the sample entities.

Table 4.2: Improvement by mNBC

Entity #	Different	Improved	Refine Rate	Accuracy
300	58	20	34.5%	87.3%
400	78	22	28.2%	86.0%
500	98	30	30.6%	86.4%

Next, we take two examples to illustrate the refinement achieved by mNBC. One entity is Gyula_Andrássy, who was a Hungarian politician. The training class chain for this entity is "Agent, Person, Politician, President" while the one generated by mNBC is "Agent, Person, Politician, PrimeMinister". After careful examination, we confirm that this person served as Prime Minister of Hungary, which validates our refinement to this entity. Another example is Berry_Berenson, who was an American actress. This entity has a simple training class chain "Agent, Person", while our generated class chain is "Agent, Person, Artist, Actor". Because the DBpedia ontology does not distinguish between female and male (i.e., there is no class "Actress"), mNBC is considered to successfully refine the classification of this entity by discovering two more specific classes "Artist" and "Actor". The two examples stand for two different types of refinement, i.e., replacement and extension. Namely, "President" is replaced by "PrimeMinister" in the former example while "Agent, Person" is extended by appending "Artist, Actor" in the latter example.

Comparison with Other Classification Methods

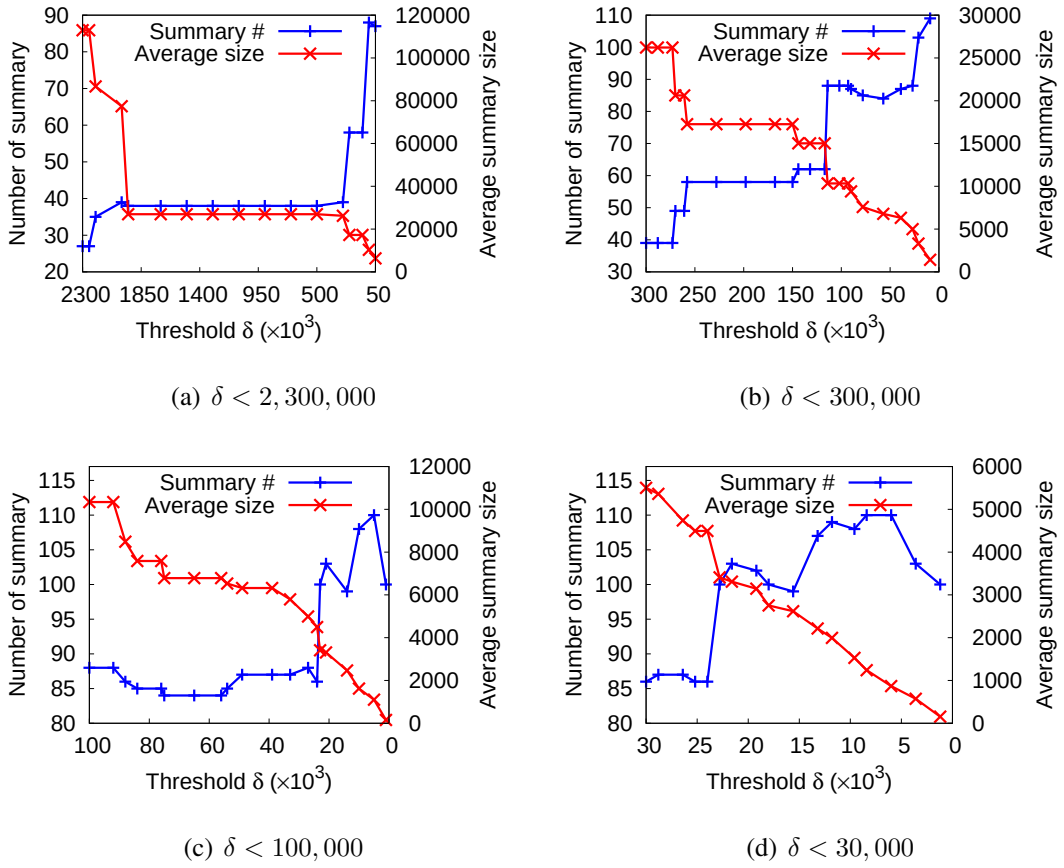
To further evaluate the performance of mNBC, we also perform a comparison study with several other popular methods, including Clus-HMC [88], RAkEL [87], MLkNN [99], HOMER and HMC [86]. Clus-HMC is a decision tree based model for hierarchical multi-label classification while HOMER and HMC both build a hierarchy of multi-label classifiers where each node is a classifier. RAkEL and MLkNN perform multi-label classification without hierarchies. Since Clus-HMC is implemented in Clus[16] and the rest are implemented in Mulan[17], we use the existing implementation of these methods directly.

Unfortunately, all these methods cannot finish the execution within 72 hours (3 days), after which we terminate them. In contrast, our mNBC model can finish the classification within 189 seconds for all entities using 10-fold cross validation. As mentioned in the introduction, the characteristics of Wikipedia infobox data (including high dimensionality, sparseness and low degree of property overlap) indeed pose great challenges for the existing classification methods. Most of them propose solutions to handling only part of the characteristics. Our mNBC model, however, can perform efficient and effective hierarchical multi-label classification by leveraging the DBpedia ontology, which helps greatly reduce the property/class searching space and improve the overall performance.

4.5.3 Evaluation of Summary Generation

In Section 4.4.2, we discussed how the threshold δ, which sets the maximum number of entities a summary can have, adjusts the granularity of generated summaries. Next we evaluate how different values of δ affect the number of summaries as well as

[16] A decision tree and rule induction system (http://dtai.cs.kuleuven.be/clus/).
[17] A library for multi-label learning (http://mulan.sourceforge.net/).



(a) $\delta < 2,300,000$

(b) $\delta < 300,000$

(c) $\delta < 100,000$

(d) $\delta < 30,000$

Figure 4.10: Number of summary vs. average summary size

the average summary size in more detail. All extracted entities are used for the generation of summaries regardless of how many properties they have and p_{min} is set to 0.9. The overall tendency is depicted in Figure 4.10(a), where the number of summaries increases and the average summary size decreases as δ reduces gradually from 2,300,000. As discussed earlier, this is because a large summary may split into more and smaller summaries following the reduction of δ. After sharp changes when δ goes down from 2,300,000 to 1,850,000, the number of summaries and average size remain stable when δ varies between 1,850,000 and 500,000. The changes become apparent again when δ is below 500,000.

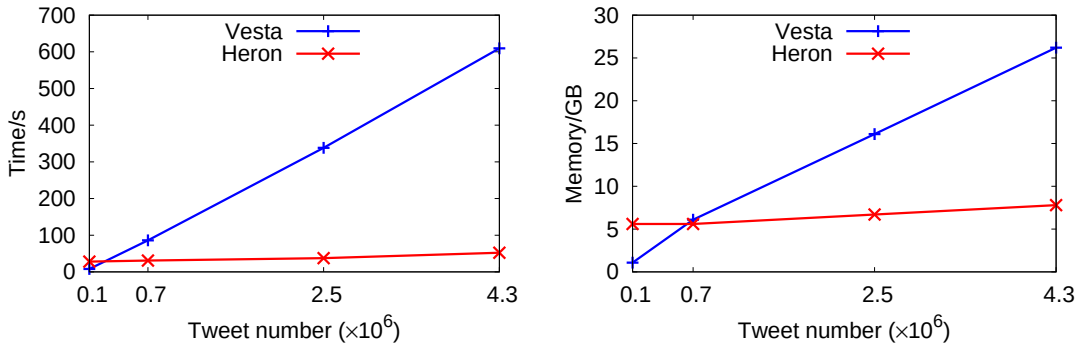
Although the overall tendency is clear, more detailed fluctuations are hidden due

to the large step value of δ in Figure 4.10(a). In light of this, we further zoom in and provide close-up changes given smaller ranges of δ in Figure 4.10(b) to 4.10(d). Figure 4.10(b) shows the tendency of the number of summaries and average size as δ decreases from 300,000, which is similar to Figure 4.10(a) with some fluctuations as well as stable stages. In Figure 4.10(c), we start to discern remarkable ups and downs for the number of summaries when δ is less than 30,000. Figure 4.10(d) further enlarges the fluctuations and we notice that the number of summaries no longer keeps increasing when δ becomes relatively small. This happens because some potential summaries (i.e., those generated when δ takes larger values) will not be generated for smaller δ. Despite the fluctuations of the number of summaries, the average summary size keeps decreasing all the time.

4.5.4 Comparison with Vesta at System Level

Vesta is a system which generates summaries of social media contents based on a biclustering approach. As reported in Section 3, Vesta outperforms several other summarization methods. Next we conduct a comparison between Heron, the summarization approach proposed in this chapter, and Vesta from several different aspects. To make the comparison feasible, we use four data sets which contain 0.1, 0.7, 2.5 and 4.3 million tweets respectively as in Section 3. Vesta has two ways to generate summaries, namely tag-based and content-based. Since the tag-based method has better performance than the content-based one, we only report the result of the tag-based method below. There are two parameters in Vesta, δ_{den} and δ_{sz}, which control the minimum density and size of a summary. We set δ_{den} and δ_{sz} to 0.8 and 3 respectively, which are consistent with the settings in Section 3. For Heron, we use all extracted entities without filtering those having a small number of properties. We also set p_{min} to 0.7 and δ to 100,000.

Figure 4.11 shows the scalability of Heron and Vesta in terms of the summary generation time and memory usage. From Figure 4.11(a), we can see that although the



(a) Generation time

(b) Memory usage

Figure 4.11: Summary generation

generation time of Vesta increases linearly as the number of tweets increases, it is quite larger than that of Heron. When the number of tweets goes up from 0.1 to 4.3 million, the generation time of Heron increases slowly, so that the increase is even negligible when compared with that of Vesta. Figure 4.11(b) also reflects the similar pattern. That is, the memory usage of Vesta increases linearly and apparently while that of Heron is almost constant. Heron outperforms Vesta in both the generation time and memory usage, which is mainly because Heron relies on the DBpedia ontology to generate summaries. The number of tweets/entities in large part affects the granularity of the generated summaries instead of bringing about high computational cost in Heron. However, a large number of tweets may lead to heavy burden to Vesta, which is sensitive to the workload.

To accelerate the generation process, Vesta further adopts a partition-and-merge (PM) scheme. The scheme puts the summary generation for smaller partitions at an offline stage, and merges summaries of smaller size from partitions to form larger summaries at runtime to speed up the system response time. Figure 4.12 shows the response time for Heron and Vesta. With the PM scheme, the response time of Vesta becomes less than that of Heron. However, considering that a set of tweets processed by Heron

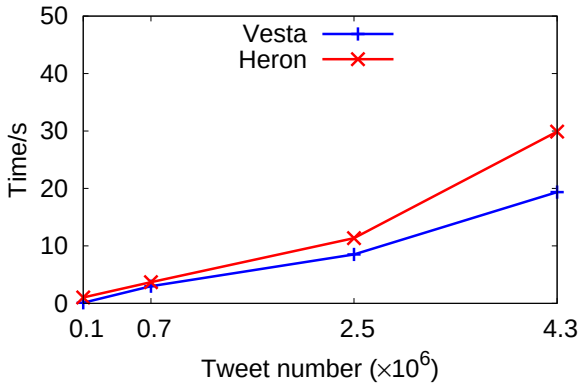


Figure 4.12: Response time

in this chapter are often related to a query string and thus do not contain as many tweets as 4.3 million (which is the approximated number of geo-coded tweets published every day [40]), the response time of Vesta and Heron will not differ too much. For instance, during a period of two months around 1.867 million tweets can be crawled for the query string "Singapore" and around 1.76 million can be crawled for "United States". What's more, the PM scheme makes Vesta a semi-realtime system because it cannot generate summaries immediately with the newly crawled tweets before they are processed at the offline stage, while Heron can process new tweets instantly and thus work in real time.

Except for the aforementioned aspects, the summaries generated by Heron and Vesta are also disparate. (1) Vesta generates summaries on the basis of individual words while Heron treats entities with multiple words as a whole. For instance, "White House" will be split into two separate keywords "White" and "House" in Vesta while Heron treats White_House as a single entity. (2) The keywords of a summary in Vesta are organized at different levels where there are no relationships among keywords at each level, while entities of a summary in Heron are organized in a sub-hierarchy presenting parent/child/sibling relationships. (3) Vesta puts keywords of a summary at different levels by doing statistical inference while Heron adds semantic relationships to entities of a summary based on the DBpedia ontology. In a word, Heron and Vesta can be supple-

mentary to each other by generating summaries from different perspectives. Interested readers are refer to Section 3 or [40] for more details about Vesta.

4.5.5 Case Study

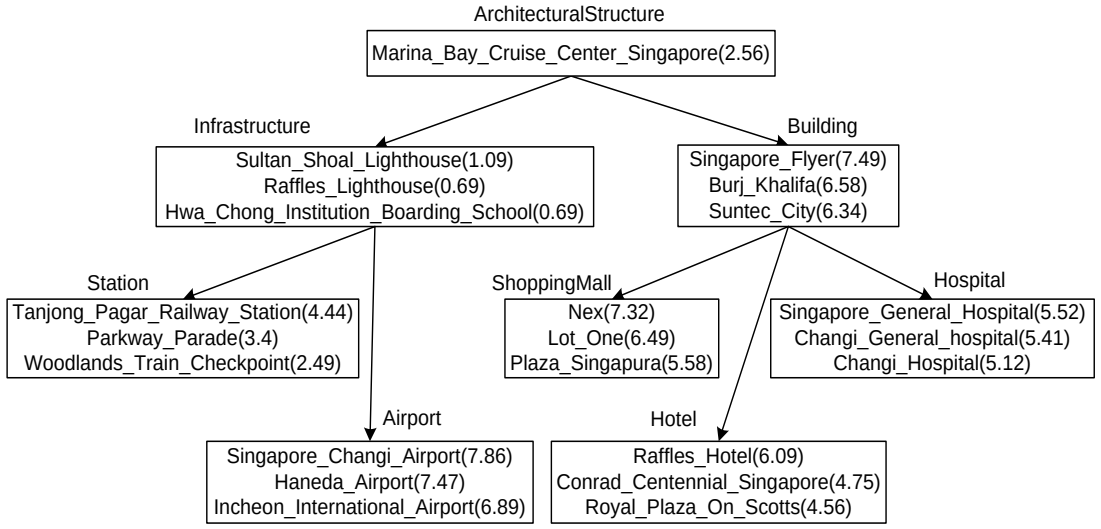


Figure 4.13: Case study: an exemplary summary

In this section, we illustrate what a real hierarchical summary looks like using an example as shown in Figure 4.13. This is one of the summaries generated from the set of tweets crawled by specifying "Singapore" as the query string. Note that the summary is a simplified version for demonstration purpose by cutting off part of the class nodes, because the original summary has twenty different class nodes which are too many for presentation.

There are eight class nodes shown in the exemplary summary. For each class node, entities are displayed in a rectangle with the corresponding class shown above it. According to Formula 4.11 (i.e., the score formula), we rank all the entities mapped to each class node and show the top three, if any, with the highest scores in Figure 4.13. The score of an entity is shown in the parentheses next to the entity's name. By looking

down from the top level to lower levels along the sub-hierarchy (the structure is called sub-hierarchy in contrast with the entire DBpedia ontology) of the summary, we note that entities in the summary are organized in different subsets. Entities in each subset are classified as the same class while different subsets are connected according to the relationships of their corresponding classes. In this manner, our proposed hierarchical summary not only groups relevant entities together but also reveals semantic relationships among them.

By ranking the entities in each class node, we also note that those closely related to the query string are often ranked higher than others. For instance, in the original summary, ten hospitals are mapped to class "Hospital". Among them, all the top nine are hospitals located in Singapore while the last one is Darent_Valley_Hospital which is a hospital in England. In other words, our proposed hierarchical summary, with the aid of the score formula, is capable of capturing the distribution of entities given any set of tweets. This further provides the opportunity for comparing the distributions of entities for tweet sets with different query strings easily.

4.6 Summary

In this chapter, we proposed Heron, a novel approach to the summarization of social media contents that generates hierarchical summaries based on the DBpedia ontology. A hierarchical summary not only groups closely related entities together, but also reveals semantic relationships by organizing subsets of entities in a hierarchical structure. To obtain high-quality summaries, we proposed an mNBC model to refine the class labels of entities. We then mapped entities with the refined classes to the DBpedia ontology, and generated summaries based on the structure of the ontology. We also proposed a score formula to rank the most relevant entities higher in each class node

of a summary. The experimental study was conducted from multiple perspectives, and demonstrated the efficiency and effectiveness of our approach.

CHAPTER 5

TRENDSPEDIA: AN INTERNET OBSERVATORY FOR ANALYZING AND VISUALIZING THE EVOLVING WEB

5.1 Overview

With an abundance of online information generated every day, users are often overwhelmed in the data ocean and have no idea where to locate useful knowledge which they are really interested in. Some social media services, such as Twitter, provide trending keywords[1] by analyzing features, e.g., hashtag frequencies[2], to help users obtain

[1] https://twitter.com/search-home
[2] http://trendsmap.com/

useful knowledge and find popular incidents. However, such preliminary analysis can lead to bias. This concern is also supported by a discovery in [47] that the majority (over 85%) of trending topics are headline news or persistent news, indicating that many other potentially interesting points are obscured by the globally hottest ones, and thus become ignored forever.

Existing research efforts (e.g., [55, 17]) often try to obtain knowledge from massive social media contents by summarizing the data, extracting trending topics, or even making predictions. Unfortunately, many people, including ordinary and corporate users, tend to have more interest in only topics or events within certain contexts, instead of the globally significant ones. Although some researchers alleviate the problem by introducing constraints, such as allowing keyword filtering [33] and focusing on contents published in specific geographic areas [61, 40], users may still find the discoveries pointless in terms of their personal preferences.

To solve the problem, we propose to bring proper context to social media contents which are streamed from the Internet. We try to index these dynamic contents via Wikipedia, a well-established online encyclopedia which has entries for a large number of entities and concepts. Organizing Internet contents around Wikipedia also creates a new way to search for content on the Internet, compared with conventional search engines such as Google[3] and Baidu[4]. Emerging effort in the same direction is also taken by Google in the form of Google Knowledge Graph[5] (cf. Figure 5.1), which allows users to browse other related entities by linking relevant entities in a graph. However, unlike Google Knowledge Graph, we deal with dynamic information related to each Wikipedia entity, and try to extract knowledge from it with analytics and visualization tools for better exploration and understanding.

[3]https://www.google.com/
[4]http://www.baidu.com/
[5]http://www.google.com/insidesearch/features/search/knowledge.html

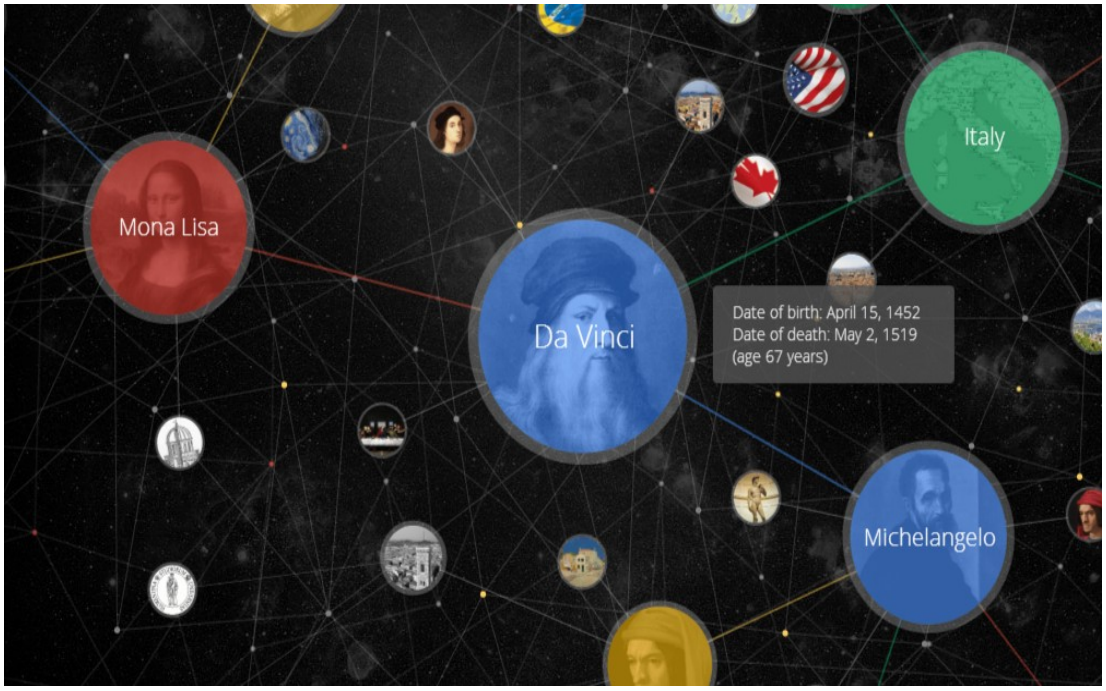


Figure 5.1: Google Knowledge Graph

Based on the aforementioned methodology, we present a novel system called Trendspedia[6] in this chapter. Trendspedia aims to provide a collaborative Internet observatory platform for users to fetch and digest the information flow on the Internet with great ease. In Trendspedia, Wikipedia articles serve as a knowledge base, so that social media contents are crawled and then routed to the relevant Wikipedia articles for further analysis.

We introduce four data analytics tools in Trendspedia. The first three tools aim to enrich each target Wikipedia entity by extracting the hottest web contents, generating summaries and emerging events respectively through an analysis of relevant social media contents. The last tool tries to build an information network that reflects the connectivity among relevant Wikipedia entities centered with the target. To enhance user experience, we visualize the analytical results so that users can explore them easily. For instance, summaries of the related social media contents of a Wikipedia entity will be

[6]http://trendspedia.com/

depicted as hierarchical tag clouds to allow users to view the summaries in an interactive manner.

By doing this, we effectively address the aforementioned challenge, enabling users not only to pinpoint useful information and knowledge they really have an interest in around Wikipedia, but also to navigate effortlessly to other closely related entities through the information network. The contents are crawled against social media services by using the entity name of a Wikipedia article as the query string for filtering. Although we currently retrieve Twitter messages, i.e., tweets, as the major social media contents, we envision that other content sources can be easily incorporated into Trendspedia for more diversified analysis.

To summarize, we make the following contributions in this chapter: (1) We build a system named Trendspedia to help users to pinpoint information and knowledge easily, according to their preferences. (2) With Trendspedia, we aim to index, organize and analyze social media contents around Wikipedia entities. (3) We implement four data analytics tools and visualize the corresponding analytical results for better exploration and understanding.

The rest of the chapter is organized as follows. The system architecture is introduced in Section 5.2. The data analytics tools are then discussed in Section 5.3. After that, a more concrete introduction to the system design and interface is presented in Section 5.4. Lastly, a summary of this chapter is provided in Section 5.5.

5.2 System Architecture

The architecture of Trendspedia is shown in Figure 5.2, which consists of three components in total, including the data storage component, the data processing component and the visualization component. The data storage component aims to store the

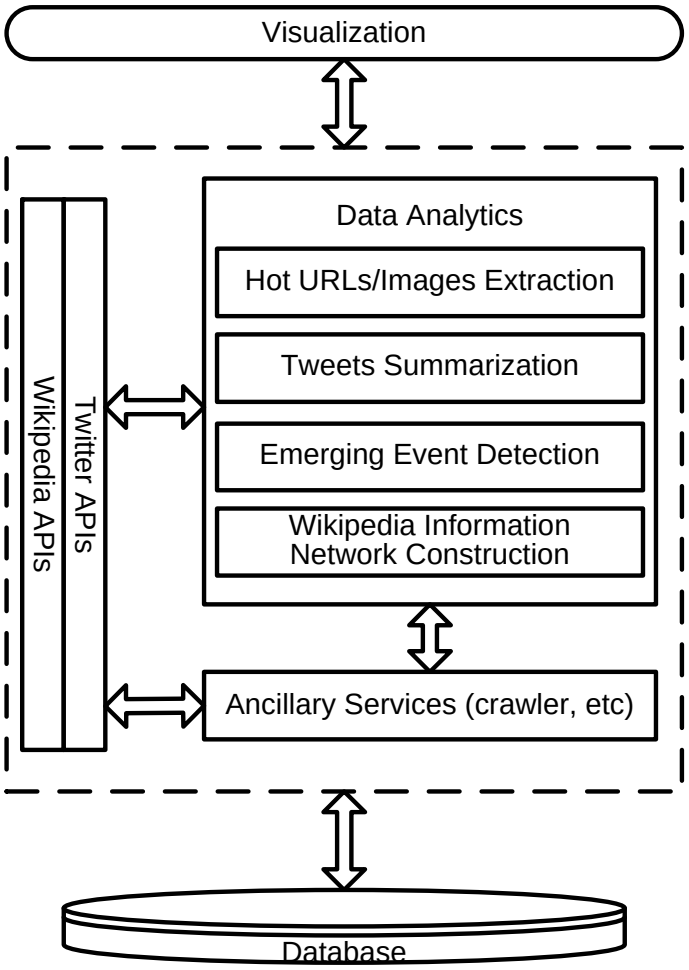


Figure 5.2: System architecture

retrieved social media data efficiently to support fast data access for the analytics tools, while the visualization component visualizes the results produced by these tools. The data processing component is described in more detail below.

The dashed box highlights the core component of Trendspedia, i.e., the data processing component, which provides ancillary services and performs different types of data analytics jobs. The ancillary services run in the background, preprocessing raw data, collecting statistics, and helping to ensure that the data analytics jobs are done properly. The major ancillary services in Trendspedia include a Twitter Crawler, a Job

Scheduler and a Tweet Analyzer. By default, the Twitter Crawler alternates to retrieve tweets containing the titles of different Wikipedia articles (i.e., entities), which have been visited by users. In addition, the Twitter Crawler also works periodically for an opened Wikipedia article, such that the more frequently an article is visited, the more tweets related to the corresponding entity are crawled. The Job Scheduler maintains a job queue so as to run multiple Twitter Crawlers simultaneously and balance the query frequencies of these crawlers to avoid potential problems which may be caused by Twitter's API rate limits[7] (rate limiting in version 1.1 of the Twitter API generally allows a limited number of requests per rate limit window, i.e., 15 minutes, for each access token). With new tweets crawled for a certain entity, the Tweet Analyzer updates statistical information about the tweets continuously.

With the aid of the ancillary services, Trendspedia implements four useful data analytics tools to enrich Wikipedia entities. The tools are designed to provide each Wikipedia entity with (1) the most relevant and hottest URLs/images, (2) summaries of related tweets, (3) recently emerging events, and (4) an information network connecting relevant Wikipedia entities. The first three tools function by analyzing tweets relevant to a target Wikipedia article while the last one works by analyzing the relationship between the target and other related Wikipedia articles.

In the next section, we introduce the technical details of these data analytics tools in Trendspedia.

[7] https://dev.twitter.com/docs/rate-limiting/1.1

5.3 Data Analytics

5.3.1 Hot URLs/Images Extraction

As a growing number of tweets are crawled and attached to a certain Wikipedia entity, popular URLs that are often mentioned in those tweets can be identified such that the web contents linked to by the URLs can be retrieved and analyzed, in turn, to enrich the corresponding Wikipedia entity. According to the statistics maintained in Trendspedia, over 70% tweets crawled by the system have at least one URL. Extracting and integrating the corresponding web contents of hot URLs will undoubtedly provide supplementary and colorful information for each Wikipedia entity.

To present hot web contents, one challenge is to avoid information duplication. Although some URLs are different, the contents of their web pages might be similar or even exactly the same. This happens very likely especially when social media users share breaking news or emergent events, such as an Apple new product launch event and an earthquake, from popular news portals. In order to estimate the popularity and remove duplicates of such URLs with identical contents, the web pages are crawled and an analysis of similarity detection is conducted. Specifically, the content of a web page is first extracted and converted to a sequence of q-grams, which are then transformed into a vector of integers. Because the vectors of different web pages are often high-dimensional and sparse, we further compress them to have shorter length but still preserve the most important features by applying the min-wise independent permutations approach [16]. After that, the compressed vectors are compared against one another according to a similarity metric, such as the cosine similarity, to group URLs linked to similar web pages together.

We then sort the groups in terms of group size in descending order, and choose one URL from each of the top-ranked groups to form hot URLs. These hot URLs are

updated from time to time when more tweets with URLs are retrieved. The first few images on the web page linked to each hot URL are also extracted and displayed as a supplement to these hot URLs.

5.3.2 Tweets Summarization

Although only relevant tweets are routed and attached to each Wikipedia entity, the continuously incoming tweets tend to discuss various aspects of the entity. This demands Trendspedia to be able to summarize those recently published tweets in order for users to easily get a multi-faceted understanding of what is going on as for each Wikipedia entity.

To this end, we adopt the formal concept analysis based summarization approach proposed in Section 3 for fast extraction of interesting summaries from a number of recent tweets. We first make use of the tweets to build a tweet-keyword matrix, where each element is either 1 or 0, indicating whether a tweet contains a keyword or not. Based on this matrix, our approach can efficiently generate a set of formal concepts. A formal concept is a sub-matrix containing a set of tweets and a set of keywords, where the keywords frequently co-occur in these tweets. The tweets in a formal concept are clustered together due to the common keywords they share, meaning that they are quite likely to discuss similar things. Thus, keywords in a formal concept naturally serve as a summary of the tweets. We select top summaries by ranking them based on the size and density of the corresponding formal concepts according to Formula 3.3, and visualize them in hierarchical tag clouds.

Our tweets summarization approach has the following characteristics: (1) *Efficient*. Empirical experiment shows our approach runs at least an order of magnitude faster than the popular topic modeling method LDA [15]. (2) *Easy for understanding*. In a formal concept, the keywords and the tweets are generated simultaneously such that

users can choose to view the relevant tweets if they want to explore more about the summary (i.e., keywords). (3) *Granularity customizable*. Similar formal concepts can be merged together to a certain extent according to a user-specified density threshold, such that the resultant summaries are extracted from tweets that are of greater cohesion or diversity. (4) *Visually interactive*. We visualize the summaries as hierarchical tag clouds, which allows users to explore the summaries interactively by zooming in/out through the tag clouds.

5.3.3 Emerging Event Detection

Another useful feature of Trendspedia is its ability of analyzing tweet streams to detect emerging events for Wikipedia entities. The event detection tool enriches an entity by filtering out meaningless Twitter messages and highlighting important emerging events happening recently. For instance, when users are browsing the Wikipedia article of "Singapore", Trendspedia augments the page by listing recent events (e.g., concerts, celebration gatherings, etc) that have happened there. Such events are not readily available in the Wikipedia article, but can be mined out from the collection of relevant tweets.

Specifically, we extract the top-k emerging events by performing temporal analysis of relevant tweets in Trendspedia. The observation is that, with the outbreak of a certain event on a Wikipedia entity, the number of relevant tweets will increase sharply. As an example, when the tragic bombing hit Boston on 15 April 2013, many Twitter users were discussing this disaster online and a large number of tweets containing the keyword "Boston" were created overnight accordingly. Therefore, given a collection of relevant tweets that span over a time period $[t_s, t_e]$, we slice the time period on a daily basis and utilize the following two criteria to quantify the importance of each slice: (1) *Popularity*, the increase of tweet number compared to that of the previous slice, mea-

suring how influential an event is; (2) *Freshness*, the time span from the slice to current time, measuring how recent an event is.

We linearly interpolates the normalizations of the above two measures to derive the score of each slice. Then, k slices with the largest scores are selected out. We consider these slices to represent k emerging events that have happened on this entity during $[t_s, t_e]$. To describe each of these k events, we further choose the top 10 words that occur most frequently in the corresponding slice.

5.3.4 Wikipedia Information Network Construction

Since Wikipedia by itself is a collection of web pages linked to each other, we can perceive it as an information network, where nodes in the network represent Wikipedia entities while links indicate interconnectivity among different nodes. Users can make use of such an information network to discover relationships of Wikipedia entities and navigate from one entity to other closely related ones.

In trendspedia, we extract and build a sub-network for each Wikipedia entity. Specifically, given a Wikipedia entity, by analyzing the content of its corresponding article we construct a two-layer directed graph. Nodes in the first layer are Wikipedia entities mentioned in the content of the Wikipedia article while those in the second layer are mentioned in the contents of Wikipedia articles of the first-layer nodes. An edge between node a and b indicates a "contains" relationship of the two entities. Note that the resultant graph may contains loops since a node might be contained in other nodes in the same and/or a different layer. The graph we construct using Wikipedia entities is similar to the web page linkage graph, thereby enabling us to run PageRank [64] to allocate weights to different nodes.

By doing this, Trendspedia provides users a weighted graph centered with an entity, the Wikipedia article of which they are reading. Although Wikipedia API can return

a list URLs of the relevant Wikipedia articles, the visualized information network in Trendspedia allows users to grasp the semantic importance and interconnectivity of relevant entities at a glance, such that it becomes much easier for them to decide which entities to explore next.

5.4 System Design and Interface

In this section, we introduce the design and interface of every aspect of the Trendspedia system in more detail, from logging in to the system to pinpointing the information regarding a specific Wikipedia entity, and from reading incoming tweets related to an entity to exploring the results of the analytics tools.

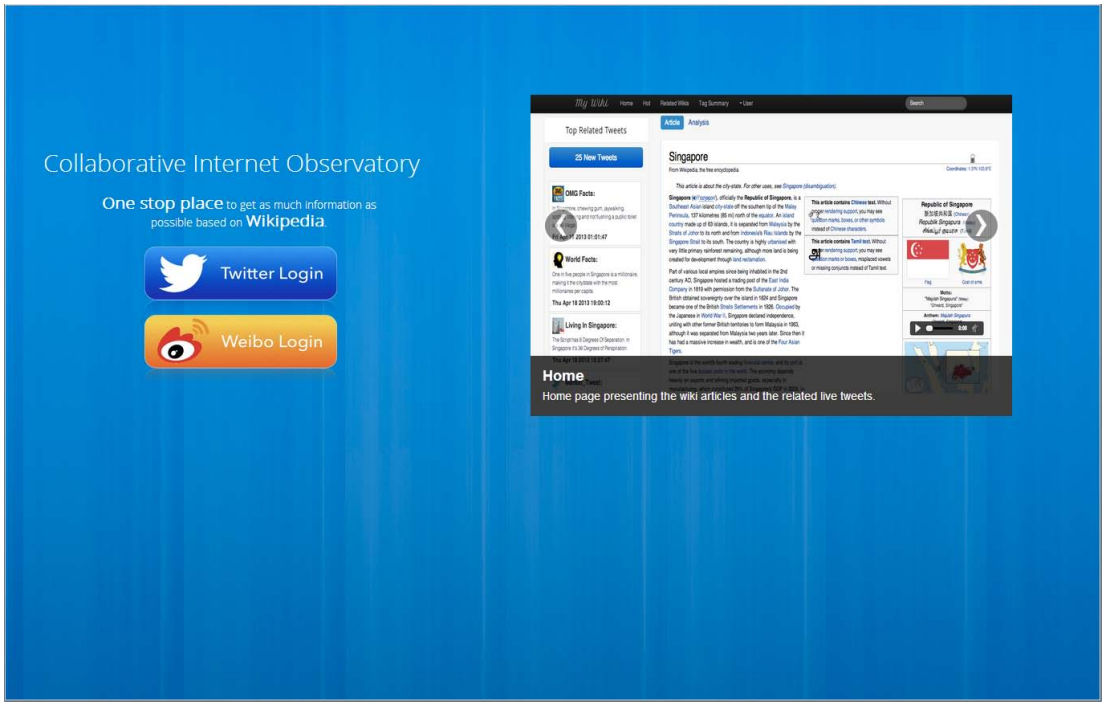


Figure 5.3: Login page

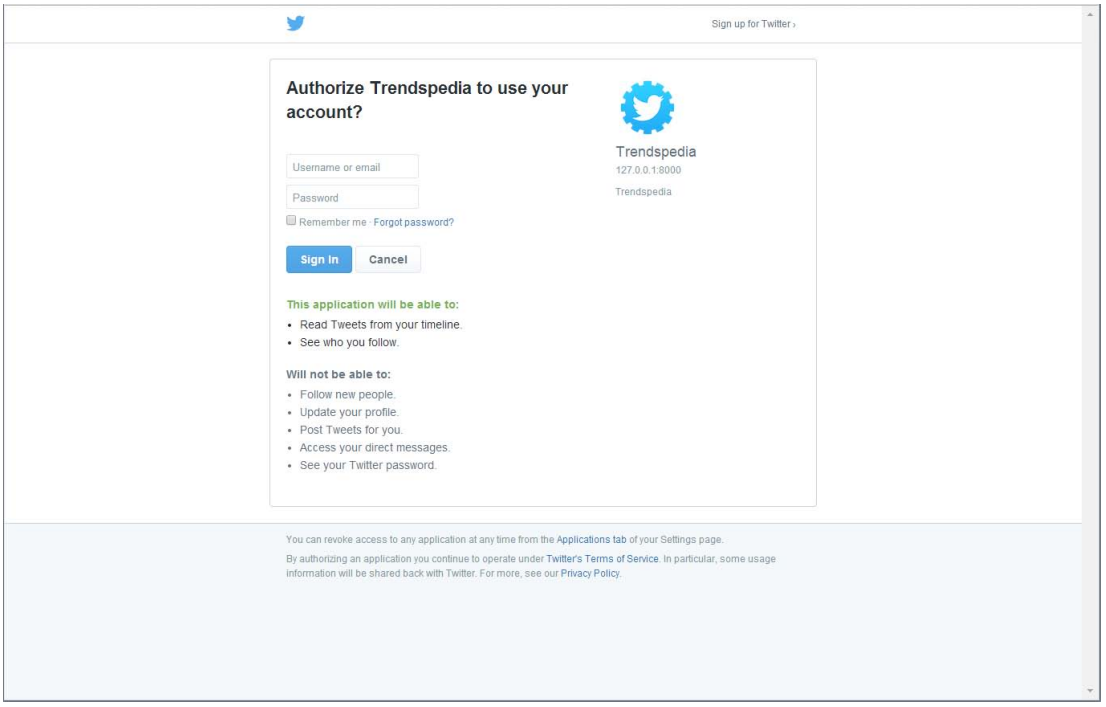


Figure 5.4: Login with Twitter account

5.4.1 System Login

Figure 5.3 displays the login page of Trendspedia. Logging in to Trendspedia is quite convenient. Users only need to use their Twitter[8] or Weibo[9] (the most popular microblogging service in China) account to log in without creating any new account in our system. Suppose users choose to log in using their Twitter account. By clicking on the "Twitter Login" button, they will be directed to Twitter's application authorization page, as shown in Figure 5.4. After inputting the Twitter account and password and clicking on the "Sign in" button, they will be directed back to Trendspedia's home page, as shown in Figure 5.5.

[8]https://www.twitter.com
[9]http://www.weibo.com

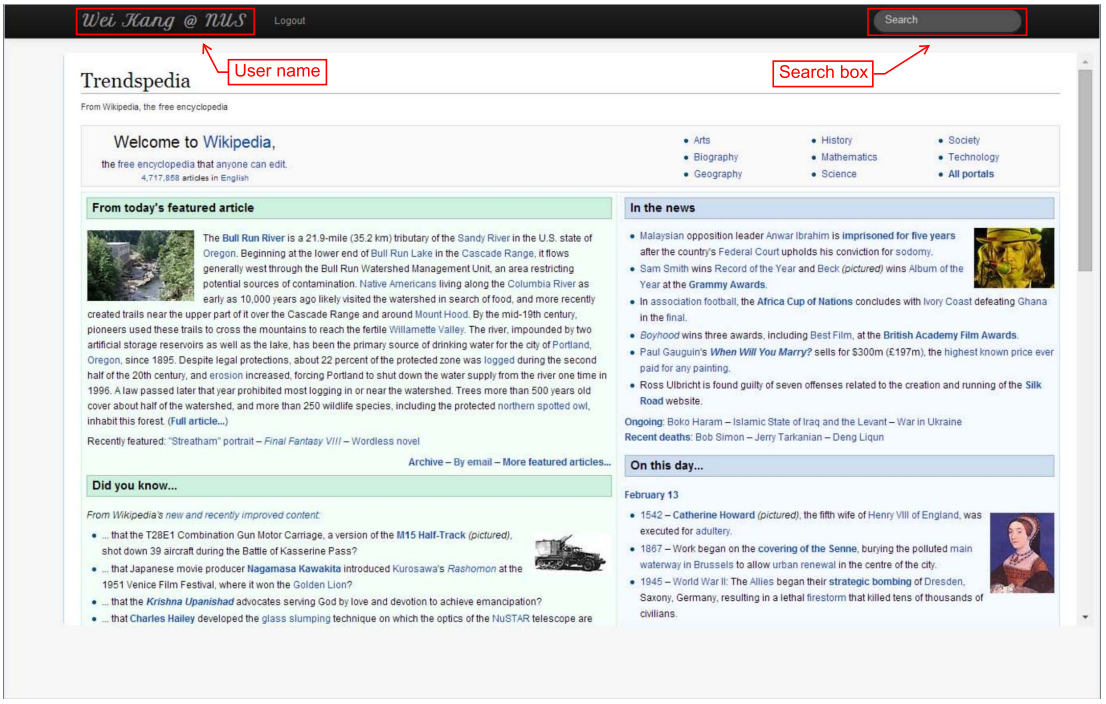


Figure 5.5: Home page after login

5.4.2 Entity Search

In Figure 5.5, in the top left-hand corner of the home page is the user name of the account used to log in to Trendspedia, while in the top right-hand corner is a search box. To pinpoint any interested Wikipedia entity and its relevant tweets as well as the analytical results, users can search for that entity in the search box. Once users key in some entity and press the "Enter" key, a list of candidate Wikipedia entities that are closely related to the user input will be displayed. For instance, if one wants to search for "Singapore", a list of relevant Wikipedia entities, such as "Singapore", "Singapore strategy" and "Singapore dollar ", as well as their corresponding descriptions will be returned, as in Figure 5.6. Users can choose from among these candidate entities and click the link "Click here" below each entity to view details of that entity.

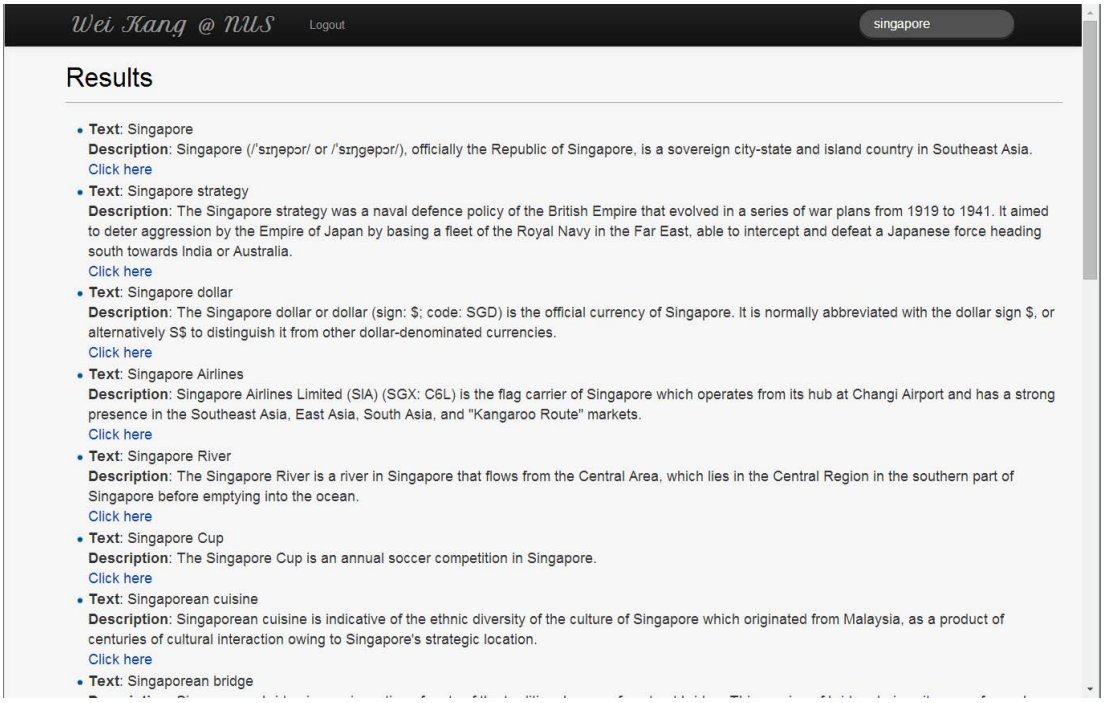


Figure 5.6: Searching results

5.4.3 Web Page of Wikipedia Entity

Suppose we open the web page of "Singapore" from the searching results in the previous figure in Trendspedia. What users will see is shown in Figure 5.7, where the right panel shows the Wikipedia article of Singapore while the left panel presents recently published tweets related to Singapore. As more relevant tweets are crawled, the number of new incoming tweets will be shown to users on the big blue button. By simply clicking on the button, users can load and read new tweets in the left panel. Once a Wikipedia article is visited, the corresponding Wikipedia entity will be added to a job queue so that more tweets related to that entity can be crawled for future analysis in Trendspedia.

Different from the home page shown in Figure 5.5, two more functional links appear on top of the web page of each specific Wikipedia entity, which are "Article" and "Analysis". Users can click on "Analysis" to open a dropdown list of different analytics

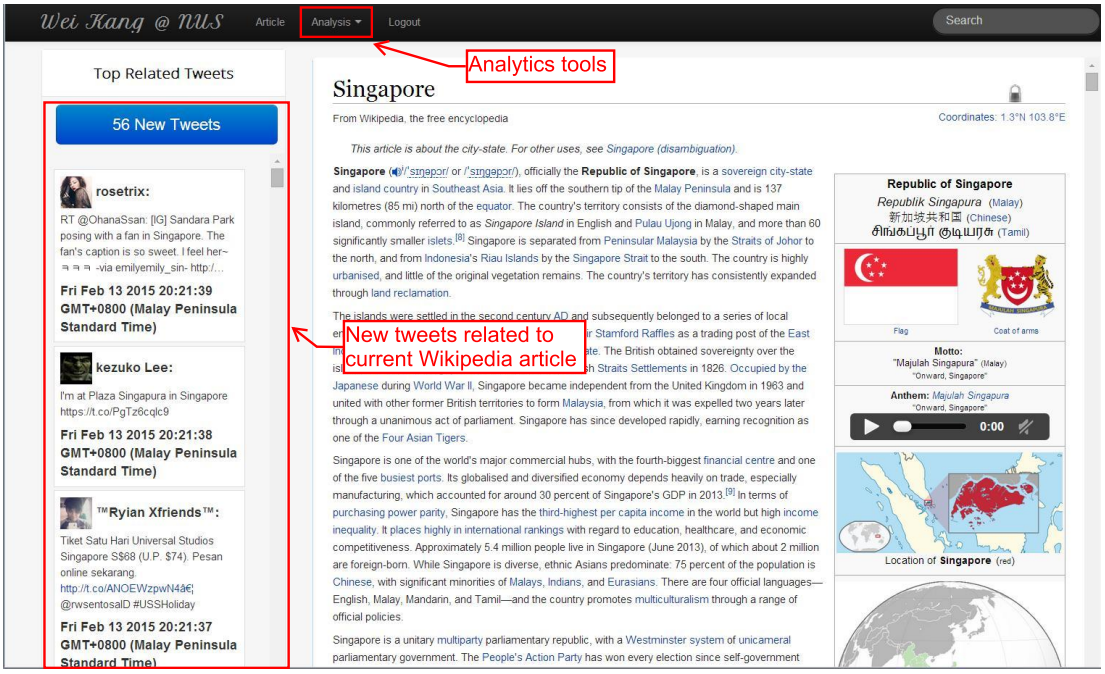


Figure 5.7: Snapshot of the "Singapore" page in Trendspedia

tools, and click again on the name of a certain analytics tool to see the visualized results produced by that analytics tool. By clicking on the "Article" link, they can switch back to the Wikipedia article.

5.4.4 Details of Analytics Tools

Clicking on a specific analytics tool in the "Analysis" dropdown list, users will be presented with the corresponding visualized results.

Figure 5.8 shows the hot URLs/images produced by the first analytics tool. For each hot URL, the title is displayed and a few sentences are excerpted from the content of the corresponding web page to provide users with a short sketch. A few images are extracted from each web page to give a more vivid impression of what is included in the web page. If interested, users can simply click on the title of a hot URL to read details of the corresponding web page.

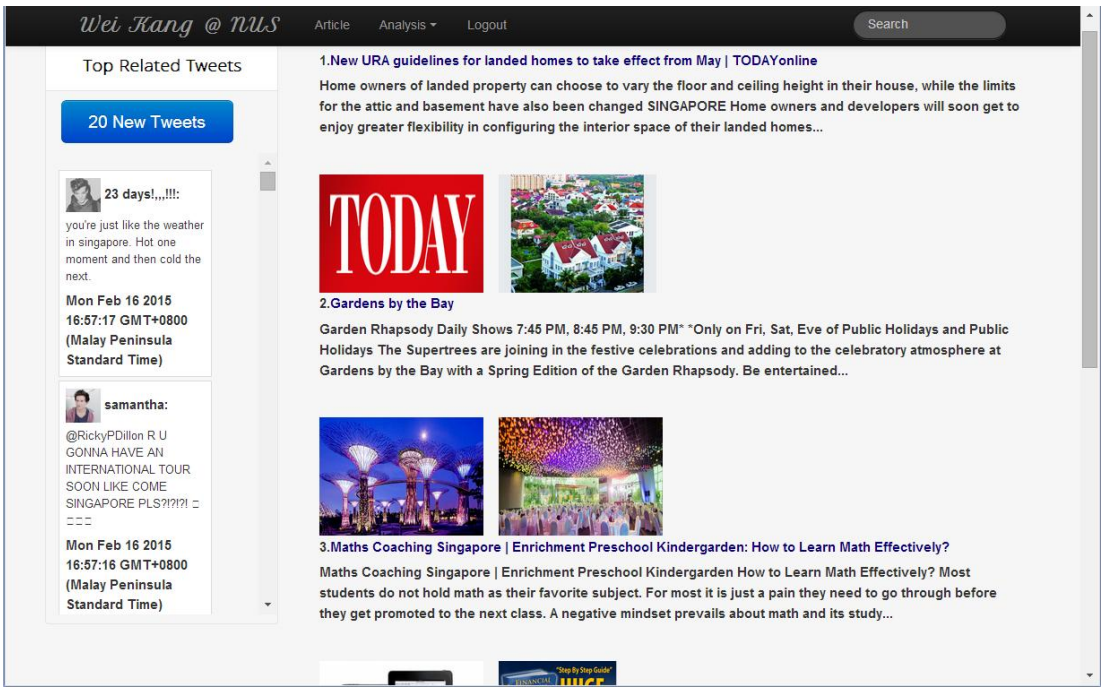


Figure 5.8: Hot URLs/images

Since a large number of tweets can be crawled for a Wikipedia entity, we generate summaries for the latest tweets so that users can grasp what is going on regarding an entity without any effort. As shown in Figure 5.9, summaries of tweets are visualized as hierarchical tag clouds. Different colors represent different summaries. In each summary, the font size of tags (keywords) indicates their generality in the set of tweets from which the summary is generated. That is, the larger the size is, the more general the tag is. Users can zoom in/out on the tag clouds interactively to view tags with more specific/general meanings in each summary. To know more details of a summary, they can read corresponding tweets by clicking on any tag in that summary, so that the tweets can slide to appear on the left side of the current web page. Clicking on different tags of the same summary will lead to different ordering of the tweets, where those containing the clicked tag will be displayed on top. Clicking on the empty area instead of any tags will hide the list of tweets.

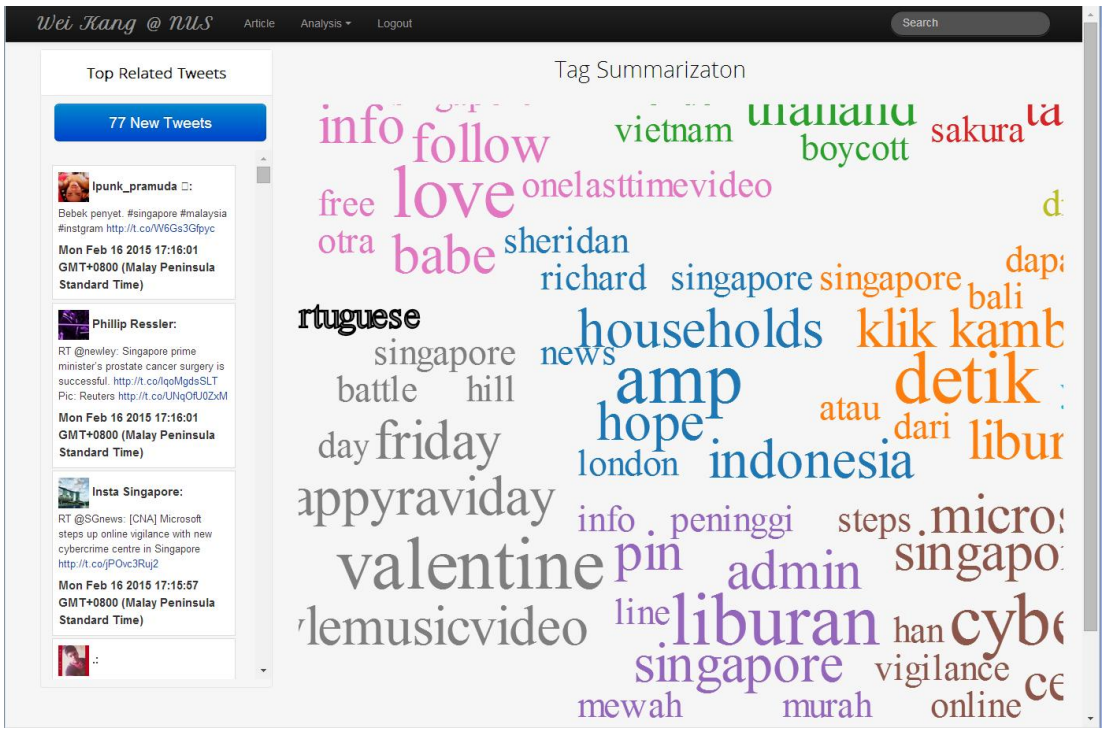


Figure 5.9: Tweets summarization

As another example, Figure 5.10 shows the summaries of tweets for Wikipedia entity "Egypt" using a different rendering scheme (e.g., different font, coloring and visualization approach), which was first discussed in [39]. These summaries were generated based on tweets crawled for "Egypt" in August 2013. As shown in Figure 5.10, the summaries, such as ("morsi", "brotherhood", "elected", "leader", ...) and ("killed", "cairo", "activist", "police", "reuter", ...), successfully capture and summarize the uprising and coup happening in Egypt in July and August 2013 from various aspects. Users may refer to Chapter 3 and [39, 40] for details of tweets summarization and hierarchical tag clouds.

Differently yet complementarily, the Emerging Event Detection tool provides a temporal perspective to exhibit recently bursty events along a timeline. Figure 5.11 shows the latest emerging events detected for the Wikipedia entity of "Singapore". Emerging events are discovered on a daily basis in Trendspedia. They are displayed as static tag

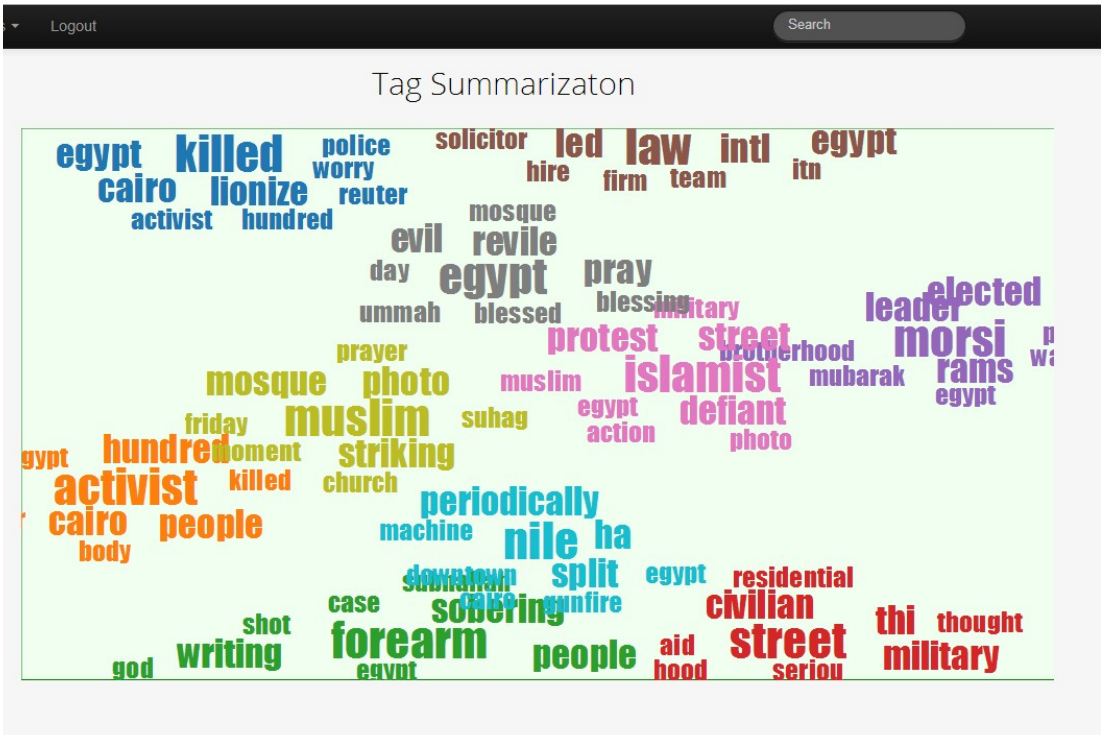


Figure 5.10: Tweets summarization for “Egypt”

clouds in chronological (descending) order, such that the latest event is shown at the top of the list. Each event consists of a set of emerging keywords, the size of which indicates the corresponding term frequency. In trendspedia, we present the top 10 keywords for every event. Note that not all days have emerging events because of our assumption that only those having a sharp increase in the number of tweets are likely to produce events.

As an example of emerging events, let’s look at the third tag cloud which captures an event happening on 21 February 2015, as shown at the bottom of Figure 5.11. The event is regarding the news that Mr Lee Kuan Yew, the first Prime Minister of Singapore, was hospitalized with severe pneumonia, which was announced by the Prime Minister’s Office of Singapore on 21 February[10]. As Singapore’s founding Prime Minister, Mr

[10]http://www.straitstimes.com/news/singapore/more-singapore-stories/story/former-singapore-pm-lee-kuan-yew-hospital-severe-pneumon

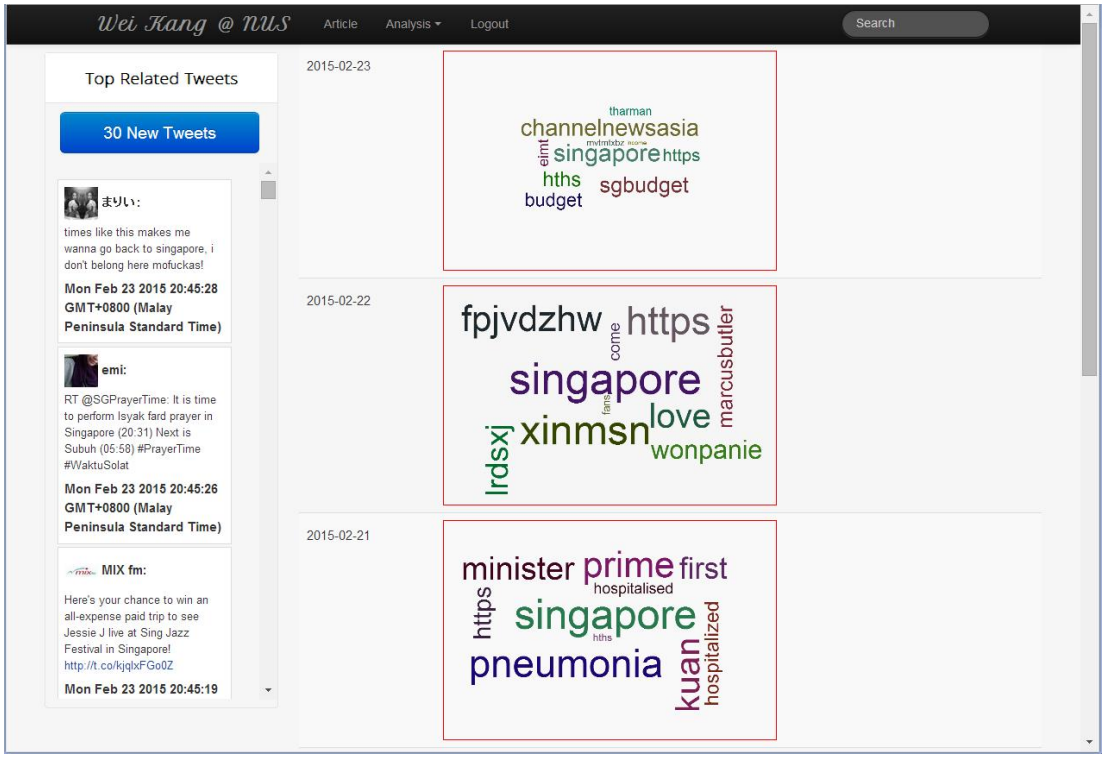


Figure 5.11: Emerging events

Lee is highly respected. Therefore, the announcement regarding his health condition undoubtedly became an emerging event on that day in Singapore, which is successfully detected by our system.

Finally, the information network centered with the target Wikipedia entity visualizes the relationships with other entities, as shown in Figure 5.12. Each node in the figure represents a Wikipedia entity. The size of a node indicates the relative term frequency of the corresponding entity in the Wikipedia article of the target entity, while the distance between the node and the central target Wikipedia entity indicates the PageRank value of the two corresponding Wikipedia entities in the information network. To avoid a mess in visualization, we do not show the edges of two connected nodes. However, hovering the mouse over a certain node can highlight other nodes which are directly connected with that node. Nodes in an information network are divided into five categories, in-

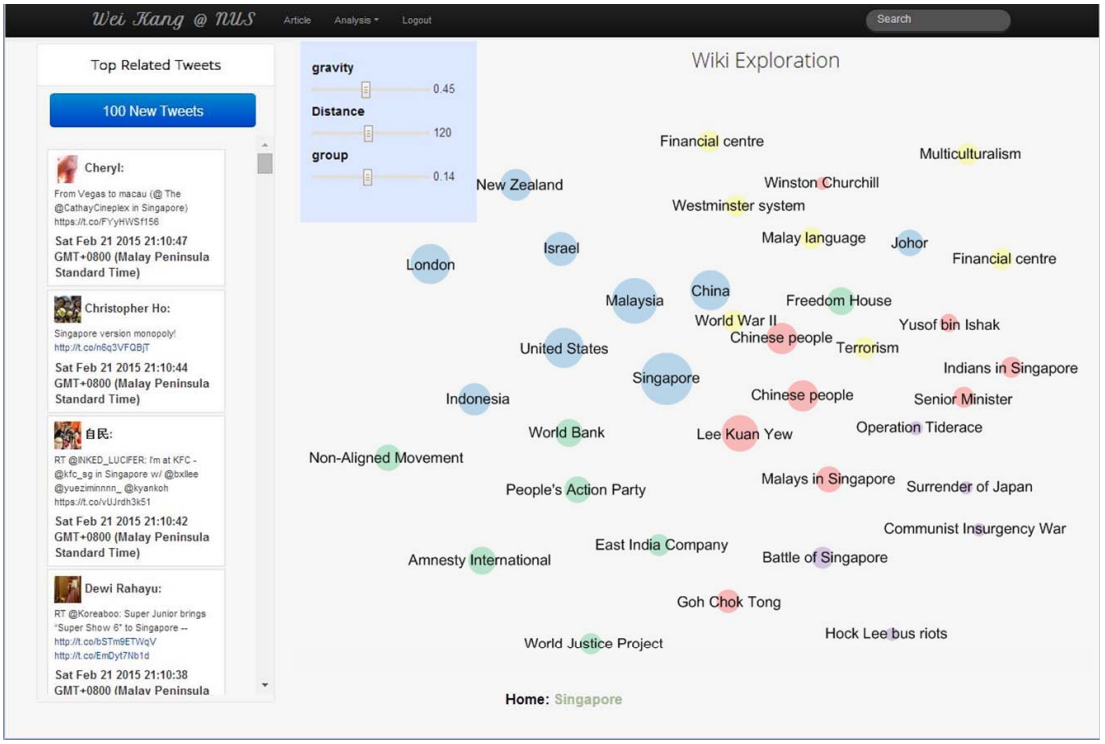


Figure 5.12: Information network

cluding people, location, organization, event and other entities. These categories are distinguished by different colors, which are red, blue, green, purple and yellow respectively.

As shown in Figure 5.12, there are three sliders in the top left-hand corner of the information network. The sliders correspond to three parameters, including *gravity*, *distance* and *group*, which are used to adjust the position and stability of the nodes in the information network. Specifically, *gravity* is used to control the force of attraction between surrounding nodes and the central target node. The larger the value of *gravity* is, the stronger the attraction force of the central target node is. The second parameter *distance* controls the relative distance among all the nodes. It works like a magnifier in that increasing the value of *distance* can be helpful for easier inspection when some nodes stay too close to one another. The last parameter *group* is used to control the intra-categorical gravity, namely the force of attraction among nodes within each category.

The larger the value of *group* is, the closer the nodes in the same category stay.

Clicking on any node leads to an expansion of the current information network by adding in more nodes which are directly connected with the clicked one. Users are also empowered to perform visualized join operation by only adding in nodes which are directly connected with a few certain nodes in current information network. They can select multiple nodes by pressing the "shift" key and clicking on the nodes that they want to choose. After that, releasing the "shift" key and then double-clicking on any empty area will lead to an expansion of the information network based on the selected nodes.

5.5 Summary

In this chapter, we presented Trendspedia, an Internet observatory platform bringing proper context to social media contents for analyzing and visualizing the web. Trendspedia tries to index, organize and analyze massive social media contents around Wikipedia entities so that users can pinpoint useful information and knowledge in terms of their preferences. Four analytics tools are adopted, and the analytical results are visualized in Trendspedia. A detailed demonstration of the system design and interface was presented, which helps guide users to explore Trendspedia step by step, and showcases how Trendspedia integrates Wikipedia and the Twitter message stream and performs data analytics as an Internet observatory.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The popularity of social media services has led to many innovations in information acquisition in modern society. A huge amount of data is generated every single day, which brings about great challenges to traditional information acquisition, integration and digestion. Therefore, there is a very urgent need to extract compact yet informative knowledge through efficient analysis of such massive amounts of data. Various types of extracted knowledge also demand an effective means of organization and management, so that ordinary users can easily obtain the information and knowledge that they are interested in.

To tackle the challenges, in this thesis we proposed two summarization approaches to the extraction of knowledge from social media contents, and presented one system for effective management of the extracted knowledge. Both these summarization approaches try to generate summaries with hierarchical structures which can be easily visualized for interactive exploration, although one focuses on the revelation of spatiotemporal knowledge and the other attempts to introduce and integrate semantics in

summarization. We also built a system as an Internet observatory platform to manage and analyze social media contents around Wikipedia, thereby enabling users to pinpoint information and extracted knowledge with great ease in accordance with their preferences. We next recapitulate the major contributions made in this thesis, and discuss some possible future research directions.

6.1 Summary and Contributions of the Thesis

Firstly, we proposed a summarization approach to the generation of summaries from spatiotemporal social media contents, so that users can discover what is happening in certain geographical regions during certain periods of time. To enable interactive exploration and better understanding of the summaries, we proposed a novel concept called hierarchical tag clouds. A hierarchical tag cloud is the visualized form of a summary, with more general tags displayed at higher levels and more specific tags displayed at lower levels. By zooming in/out through hierarchical tag clouds, users are able to interactively explore and understand the summaries at different levels of abstraction. To support this, we proposed an efficient summarization approach by biclustering social media contents. We also extended it to a partition-and-merge scheme to enhance the scalability.

Secondly, we proposed to generate a new type of summary, namely the hierarchical summary, which is especially designed to introduce and explore semantics. A hierarchical summary consists of a set of closely related Wikipedia entities extracted from social media contents. Furthermore, according to their class labels, the entities are divided into different subsets, each of which is mapped to a class node in a sub-hierarchy of the DBpedia ontology. As a result, a hierarchical summary not only consists of closely related entities, but also possesses a hierarchical structure which connects the subsets

of entities and reveals the semantic meanings and subsumptive relationships among the subsets of entities. To reduce the propagation of inaccuracy in Wikipedia, we proposed a model named multi-level Naive Bayes Classifiers to refine the classes of entities before mapping them onto the DBpedia ontology. Since a large number of entities might be mapped to a single class, we also introduced a ranking procedure to select the most important and relevant entities in each subset. The inherent structure of hierarchical summaries enables the easy visualization of them in hierarchical tag clouds.

Finally, we presented a novel system which brings proper context to continuously incoming social media contents, such that massive information can be indexed, organized and analyzed around Wikipedia entities. Four analytics tools are employed in the system. The first three tools aim to enrich Wikipedia entities by analyzing the relevant social media contents, while the fourth one builds an information network among the most relevant Wikipedia entities. With the assistance of this system, users are empowered to pinpoint valuable information and knowledge they are interested in, as well as to navigate to other closely related entities through the information network for further exploration.

6.2 Future Directions

In this thesis, we proposed to interactively explore the summaries by zooming in or out through the hierarchical tag clouds. During the process, the tags are displayed or hidden at different levels while the corresponding summaries actually remain the same throughout. This is because the summaries are fixed and will never change once they are generated for visualization. One possible improvement might be enabling summaries to update themselves adaptively when users zoom in or out over the map. In other words, summaries can be generated based on which part of the map is currently displayed,

and can expand or shrink automatically and adaptively whenever the map is panned or zoomed. The summary updates will be triggered because the set of spatiotemporal social media contents change when the map is changed. Accordingly, the summary updates will be reflected and visualized instantly in the hierarchical tag clouds. This new type of visualization will inevitably lead to the generation of summaries on the fly, instead of the pre-computation of them as done in Section 3. Possible operations may include splitting, merging and regenerating summaries when the map is zoomed in/out and panned. Although this visualization results in higher computational costs and demands more sophisticated summarization approaches, it is able to present extracted information adaptively and thus can greatly enhance the understanding of knowledge.

To introduce semantics, we proposed to generate hierarchical summaries which are designed to reveal semantic meanings and relationships among subsets in each summary. Before generating the summaries, we group entities according to the classes to which they belong in the DBpedia ontology. Since this ontology has only a few hundred classes, to generate summaries reflecting more delicate semantic relationships, an alternative is to use a more complex and well-defined ontology, such as YAGO[1] which combines the taxonomy of WordNet and the Wikipedia category system, and thus covers over 350,000 classes. Given so many classes, however, a big challenge to hierarchical summarization might be the sparseness of entities and the complexity of the relationships among the classes. Besides the ontology based summarization, other types of semantics can also be introduced, such as sentiment. Many algorithms have been proposed for sentiment analysis and opinion mining [44, 66, 67], some of which focus on sentiment analysis of social media contents [62, 65, 1]. Introducing sentiment analysis to the hierarchical summarization of social media contents can reveal fine-grained categories of various sentiments, instead of merely "thumbs-up" and "thumbs-down", or

[1] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

"happy" and "upset".

To effectively integrate and manage massive amounts of information and extracted knowledge, we built a platform to index and analyze social media contents around Wikipedia entities. The social media contents mentioned in this thesis are mainly textual microblogs. Apart from this, many other types of posts published in social media services, such as images and videos, can be integrated together to enrich the discoveries and derived knowledge from various aspects. Possible challenges in this respect include the linkage among different types of posts/entities, the integration of various sources, and user-friendly visualization.

BIBLIOGRAPHY

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38. Association for Computational Linguistics, 2011.

[2] James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 37–45, 1998.

[3] Mohamed Aly. Survey on multiclass classification methods. *Neural Networks*, pages 1–9, 2005.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735, 2007.

[5] Roja Bandari, Sitaram Asur, and Bernardo A. Huberman. The pulse of news in social media: Forecasting popularity. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[6] Nilanjan Banerjee, Dipanjan Chakraborty, Koustuv Dasgupta, Sumit Mittal, Anupam Joshi, Seema Nagar, Angshu Rai, and Sameer Madan. User interests in social media sites: an exploration with micro-blogs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 1823–1826, 2009.

[7] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121, 1999.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), 1975.

[9] Wei Bi and James Kwok. Efficient multi-label classification with many labels. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 405–413, 2013.

[10] Kai Bielenberg. Groups in social software: Utilizing tagging to integrate individual contexts for social navigation. *Master Thesis*, 2005.

[11] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

[12] David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2), 2010.

[13] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 17–24, 2003.

[14] David M. Blei and Michael I. Jordan. Modeling annotated data. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 127–134, 2003.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[16] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[17] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, pages 4:1–4:10. ACM, 2010.

[18] Junghoon Chae, Dennis Thom, Harald Bosch, Yun Jang, Ross Maciejewski, David S Ebert, and Thomas Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *2012 IEEE Conference on Visual Analytics Science and Technology, VAST 2012, Seattle, WA, USA, October 14-19, 2012*, pages 143–152, 2012.

[19] Deepayan Chakrabarti and Kunal Punera. Event summarization using tweets. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

[20] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, August 19-23, 2000, La Jolla / San Diego, CA, USA*, pages 93–103, 2000.

[21] Dongjun Chung and Sunduz Keles. Sparse partial least squares classification for high dimensional data. *Statistical applications in genetics and molecular biology*, 9(1), 2010.

[22] Bing Tian Dai, Ee-Peng Lim, and Philips Kokoh Prasetyo. Topic discovery from tweet replies. In *MLG: The Workshop on Mining and Learning with Graphs*, 2012.

[23] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 89–98, 2003.

[24] Micah Dubinko, Ravi Kumar, Joseph Magnani, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Visualizing tags over time. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 193–202, 2006.

[25] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22(1):457–479, 2004.

[26] Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[27] Bernhard Ganter, Rudolf Wille, and Rudolf Wille. *Formal concept analysis*, volume 284. Springer Berlin, 1999.

[28] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6(6):721–741, 1984.

[29] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott. *Continuous lattices and domains*, volume 93. Cambridge University Press, 2003.

[30] Dmitry Gnatyshak, Dmitry I. Ignatov, Alexander Semenov, and Jonas Poelmans. Gaining insight in social networks with biclustering and triclustering. In *Perspectives in Business Informatics Research - 11th International Conference, BIR 2012, Nizhny Novgorod, Russia, September 24-26, 2012. Proceedings*, pages 162–171, 2012.

[31] Yihong Gong and Xin Liu. Generic text summarization using relevance measure and latent semantic analysis. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 19–25, 2001.

[32] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 537–544, 2004.

[33] Adrien Guille, Cécile Favre, Hakim Hacid, and Djamel A. Zighed. Sondy: an open source platform for social dynamics mining and analysis. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 1005–1008, 2013.

[34] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan kaufmann, 2006.

[35] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: A graph-based method. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 765–774, 2011.

[36] David J Hand and Keming Yu. Idiot's bayes–not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[37] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[38] Carlos Nascimento Silla Jr. and Alex Alves Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[39] Wei Kang, Anthony K. H. Tung, Wei Chen, Xinyu Li, Qiyue Song, Chao Zhang, Feng Zhao, and Xiajuan Zhou. Trendspedia: An internet observatory for analyzing and visualizing the evolving web. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1206–1209, 2014.

[40] Wei Kang, Anthony K. H. Tung, Feng Zhao, and Xinyu Li. Interactive hierarchical tag clouds for summarizing spatiotemporal social contents. In *IEEE 30th*

International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014, pages 868–879, 2014.

[41] Owen Kaser and Daniel Lemire. Tag-cloud drawing: Algorithms for cloud visualization. *CoRR*, abs/cs/0703109, 2007.

[42] Mitchell C Kerman, Wei Jiang, Alan F Blumberg, and Samuel E Buttrey. Event detection challenges, methods, and applications in natural and artificial systems. Technical report, DTIC Document, 2009.

[43] Mahboob Alam Khalid, Valentin Jijkoun, and Maarten de Rijke. The impact of named entity normalization on information retrieval for question answering. In *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*, page 705–710. Springer, 2008.

[44] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*. Association for Computational Linguistics, 2004.

[45] Shailesh Kumar, Joydeep Ghosh, and Melba M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Anal. Appl.*, 5(2):210–220, 2002.

[46] Byron Yu-Lin Kuo, Thomas Hentrich, Benjamin M. Good, and Mark D. Wilkinson. Tag clouds for summarizing web search results. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 1203–1204, 2007.

[47] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue B. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 591–600, 2010.

[48] Helge Langseth and Thomas D Nielsen. Classification using hierarchical naive bayes models. *Machine Learning*, 63(2):135–159, 2006.

[49] Ryong Lee and Kazutoshi Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *Proceedings of the 2010 International Workshop on Location Based Social Networks, LBSN 2010, November 2, 2010, San Jose, CA, USA, Proceedings*, pages 1–10, 2010.

[50] Chenliang Li, Aixin Sun, and Anwitaman Datta. Twevent: segment-based event detection from tweets. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 155–164, 2012.

[51] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 375–384, 2009.

[52] Le Luo and Li Li. Defining and evaluating classification algorithm for high-dimensional data based on latent topics. *PLoS ONE*, 9(1):e82119, 01 2014.

[53] Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. Topic-driven reader comments summarization. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 265–274, 2012.

[54] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1):24–45, 2004.

[55] Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 1155–1158, 2010.

[56] Andrew McCallum, Xuerui Wang, and Andrés Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *J. Artif. Intell. Res. (JAIR)*, 30:249–272, 2007.

[57] Pablo N Mendes, Max Jakob, and Christian Bizer. Dbpedia: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012), Istanbul, Turkey, May 23-25, 2012*, pages 1813–1817, 2012.

[58] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, pages 1–8, 2011.

[59] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 233–242, 2007.

[60] David N. Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management,*

CIKM 2008, Napa Valley, California, USA, October 26-30, 2008, pages 509–518, 2008.

[61] Mor Naaman, Hila Becker, and Luis Gravano. Hip and trendy: Characterizing emerging trends on twitter. *Journal of the Association for Information Science and Technology (JASIST)*, 62(5):902–918, 2011.

[62] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*, 2010.

[63] Brendan O'Connor, Michel Krieger, and David Ahn. Tweetmotif: Exploratory search and topic summarization for twitter. In *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*, 2010.

[64] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[65] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*, 2010.

[66] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July,*

2004, Barcelona, Spain., pages 271–278. Association for Computational Linguistics, 2004.

[67] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[68] M. Cristina Pattuelli and Sara Rubinow. The knowledge organization of dbpedia: A case study. *Journal of Documentation*, 69(6):762–772, 2013.

[69] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 181–189, 2010.

[70] Jonas Poelmans, Paul Elzinga, Stijn Viaene, and Guido Dedene. Formal concept analysis in knowledge discovery: A survey. In *Conceptual Structures: From Information to Intelligence, 18th International Conference on Conceptual Structures, ICCS 2010, Kuching, Sarawak, Malaysia, July 26-30, 2010. Proceedings*, pages 139–153, 2010.

[71] Daniel Ramage, Susan T. Dumais, and Daniel J. Liebling. Characterizing microblogs with topic models. In *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*, 2010.

[72] Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technolo-*

gies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA, pages 1375–1384, 2011.

[73] Jesse Read, Bernhard Pfahringer, and Geoffrey Holmes. Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 995–1000, 2008.

[74] Manjeet Rege, Ming Dong, and Farshad Fotouhi. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 532–541, 2006.

[75] Jiangtao Ren, Sau Dan Lee, Xianlu Chen, Ben Kao, Reynold Cheng, and David Cheung. Naive bayes classification of uncertain data. In *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 944–949, 2009.

[76] Zhaochun Ren, Shangsong Liang, Edgar Meij, and Maarten de Rijke. Personalized time-aware tweets summarization. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 513–522, 2013.

[77] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[78] A. W. Rivadeneira, Daniel M. Gruen, Michael J. Muller, and David R. Millen. Getting our head in the clouds: toward evaluation studies of tagclouds. In *Pro-*

ceedings of the 2007 Conference on Human Factors in Computing Systems, CHI 2007, San Jose, California, USA, April 28 - May 3, 2007, pages 995–998, 2007.

[79] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 851–860, 2010.

[80] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In *Proceedings of the Third International Conference on Weblogs and Social Media, ICWSM 2009, San Jose, California, USA, May 17-20, 2009*, 2009.

[81] Lidan Shou, Zhenhua Wang, Ke Chen, and Gang Chen. Sumblr: continuous summarization of evolving tweet streams. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 533–542, 2013.

[82] James Sinclair and Michael Cardew-Hall. The folksonomy tag cloud: when is it useful? *J. Information Science*, 34(1):15–29, 2008.

[83] Yangqiu Song, Shimei Pan, Shixia Liu, Furu Wei, Michelle X. Zhou, and Weihong Qian. Constrained coclustering for textual documents. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[84] Xuning Tang and Christopher C Yang. Tut: a statistical model for detecting trends, topics and user interests in social media. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 972–981, 2012.

[85] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[86] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD08)*, pages 30–44, 2008.

[87] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE Trans. on Knowl. and Data Eng.*, 23(7):1079–1089, July 2011.

[88] Celine Vens, Jan Struyf, Leander Schietgat, Saso Dzeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[89] Xiaojun Wan and Jianwu Yang. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 299–306, 2008.

[90] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 697–702, 2007.

[91] Zheng Xiang and Ulrike Gretzel. Role of social media in online travel information search. *Tourism management*, 31(2):179–188, 2010.

[92] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 745–754, 2011.

[93] Jiong Yang, Haixun Wang, Wei Wang, and Philip S. Yu. Enhanced biclustering on expression data. In *3rd IEEE International Symposium on BioInformatics and BioEngineering (BIBE 2003), 10-12 March 2003, Bethesda, MD, USA*, pages 321–327, 2003.

[94] Jiong Yang, Wei Wang, Haixun Wang, and Philip S. Yu. delta-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, pages 517–528, 2002.

[95] Yiming Yang, Thomas Pierce, and Jaime G. Carbonell. A study of retrospective and on-line event detection. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 28–36, 1998.

[96] Dave Yates and Scott Paquette. Emergency knowledge management and social media technologies: A case study of the 2010 haitian earthquake. *International Journal of Information Management*, 31(1):6–13, 2011.

[97] Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. Multi-document summarization by maximizing informative content-words. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1776–1782, 2007.

[98] Liyang Yu. *A Developers Guide to the Semantic Web*. Springer Berlin Heidelberg, 2011.

[99] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[100] Qiankun Zhao, Prasenjit Mitra, and Bi Chen. Temporal and information flow based event detection from social text streams. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1501–1506, 2007.

[101] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 336–345, 2003.

[102] Arkaitz Zubiaga, Damiano Spina, Enrique Amigó, and Julio Gonzalo. Towards real-time summarization of scheduled events from twitter streams. In *23rd ACM Conference on Hypertext and Social Media, HT '12, Milwaukee, WI, USA, June 25-28, 2012*, pages 319–320, 2012.